

tanulmányok **140/1983**

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest



MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓINTÉZETE

OPERATIONS RESEARCH SOFTWARE DESCRIPTIONS
(VOLUME 1)

Edited by

ANDRÁS PREKOPA and GERZSON KÉRI

A kiadásért felelős:

DR VÁMOS TIBOR

Főosztályvezető:

PRÉKOPA ANDRÁS

ISBN 963 311 149 8

ISSN 0324-2951

P R E F A C E

This volume contains 9 papers prepared within the framework of the activity of Working Group No 7 KNVVT (Komissija Naučnye Voprosy Vyčislitelnoi Tehniki, in English: Committee of the Scientific Problems of Computer Science). It will soon be followed by another similar collection of publications. One of the several working groups of KNVVT is Working Group No. 7 (WG-7) which has been founded in order to develop program packages and other software products in the field of operations research.

Teams from the following countries belong to WG-7: Bulgaria, Czechoslovakia, German Democratic Republic, Hungary, Poland, Roumania and the Soviet Union. Chairman of WG-7 is A. PREKOPA (Hungary) and the secretary is Yu. EVTUSHENKO (Soviet Union).

WG-7 started its activity in 1973 and terminates in 1983. There were 10 sessions organized so far in 4 countries by turns.

The majority of the papers of this collection deal with nonlinear programming. Some other subjects covered are: linear programming, transportation problems, network flows, problems of optimal control, discrete optimization.

A wide range of practical applications are also covered by the papers given in this volume, such as

- economic applications,*
- applications in sociology, biology and medicine,*
- problems of taxonomy,*
- problems of "training by tutor",*
- problems of industrial quality control,*
- forecasting and prediction problems,*
- classification and typology,*
- production location of homogeneous products,*
- highway engineering scheduling,*

- other engineering problems
(e.g. optimization of chemical reactor systems),
- computer aided design of engineering systems,
- equipment mounting in computing centres etc.

In the first paper of this volume I. EREMIN, G. KORNILOVA, L. KOROTAEVA, M. KOSTINA, N. GLEZER, M. HRIPUN, K. SUNDUKOVA, L. POPOV and V. TOMILOVA describe a program package that solves flow problems and operative planning problems. In the paper by Yu. EVTUSHENKO, E. VESELOV and V. MAZURIK an interactive optimization system is described which solves unconstrained and nonlinear optimization problems, and problems of optimal control. The program package treated in the paper by V. MALKOV, V. SKOKOV and B. CHERKASSKIY was worked out for a wide range of optimization problems such as linear, quadratic, nonlinear programming and transportation problems. An interesting type of the latter is the three index transportation problem with planar sums. Vl. MAZUROV and V. KAZANTZEV present a paper about the program package "Kvazar" for the analysis of systems of inequality constraints, which are not necessarily consistent. This package has proved to be very useful in applications to medical diagnosis, first of all for cases of cerebral diseases. The package named DISPRO described in the paper of V. MIKHALEVICH, I. SERGIENKO, T. LEBEDEVA, V. ROSHCIN, A. STUKALO, V. TRUBIN and N. SHOR solves a wide range of general and special problems of discrete optimization. The two papers about the packages NLOP-1 and NLOP-2 by G. CHRISTOV, M. IVANCHEV, R. KALTINSKA, Z. KARAMITEVA, W. KJUCHUKOVA, J. MITEV, A. DONCHEV, Z. NEDEVA, B. PARAKOZOVA and W. VELIOV and also the paper about the package NLOPT by C. RICHTER contain descriptions of libraries of nonlinear programming software units. As an interesting application, the latter mentions the optimization of the Williams-Otto-Plant in Leuna-Merseburg. In the last paper of this volume G. KÉRI, É. KOMÁROMI and P. SZ. TURCHÁNYI describe some of their computer programs for solving the classical transportation problem and the transportation problem with a piecewise linear convex objective function.

*Finally we express our thanks to every contributor of
this collection.*

THE EDITORS

CONTENTS

	Page
<i>I.I. Eremin, G.F. Kornilova, L.T. Korotaeva et al.:</i> Structure and organization of the OPTIMA-2 optimization package	9
<i>Yu. Evtushenko, E. Veselov, V. Mazurik:</i> Dialogue System for optimization	23
<i>U.H. Malkov, V.A. Skokov, B.V. Cherkasskiy:</i> Procedures for optimizing economic models	35
<i>Vl. Mazurov, V.S. Kazantzev:</i> The package of applied programs "KVAZAR" for analysis of inconsistent constraints through committee constructions	51
<i>V.S. Mikhalevich, I.V. Sergienko, T.T. Lebedeva et al.:</i> On program package DISPRO for solution of discrete optimization problems	65
<i>G. Christov, M. Ivanchev, R. Kaltinska et al.:</i> Program-package NLOP-1 for nonlinear constrained optimization	89
<i>G. Christov, A. Donchev, M. Ivanchev et al.:</i> Program package NLOP-2 for nonlinear optimization	99
<i>C. Richter:</i> On a software package for solving nonlinear programming problems	109
<i>G. Kéri, É. Komáromi, P.Sz. Turchányi:</i> Procedures for the solution of transportation type problems	133

STRUCTURE AND ORGANIZATION OF THE OPTIMA-2 OPTIMIZATION PACKAGE

I.I.Eremin, G.F.Kornilova, L.T.Korotaeva,
M.A.Kostina, N.N.Glezer, M.S.Hripun, K.A.Sundukova,
L.D.Popov, V.G.Tomilova

(Sverdlovsk, USSR)

The OPTIMA-2 software package has been developed in the Institute of Mathematics and Mechanics of the Urals Scientific Center of the USSR Academy of Science and can be used by specialists engaged with problems, concerning the application of modern economical and mathematical methods and calculating machines to the National Economy planning and control. The package was handed in the Fund of Algorithms and Computer Programs attached to the Computation Center of the USSR Academy of Science.

The paper describes the package principal capabilities and operating environment, in which it functionates.

1. THE PACKAGE GENERAL DESCRIPTION

Solving the real-life flowing and operative planning problems reduces to the necessity of the resort to the overall economical and mathematical investigation of these problems and carrying out a number of laborious computational experiments. It requires multiple reiteration of the computational procedures, starting with the formulation of the problem and terminating with more accurate definition of the model, until the acceptable solution is being obtained [1,2]. Multiformality of models and ways in finding a compromise solution causes multiple modifications of the chosen computational algorithm and corresponding set of computer programs. Such kind of investigations essentially impedes employment of the optimal planning methods and models in their standard form, i.e. in the form,

not provided with software, developed enough to enable prompt respond on the base of automated computer programs to the variations of the productional and corresponding computational processes.

The software of these investigations can be created only on the base of qualitative new software engineering - software packages [3,4].

In practice the computational process falls into a sequence of calculational procedures, divided by periods of time of different duration, depending on the reaction interval Δ , which is equal to the unit of time (quarter, month, etc.). Duration of the reaction interval is just the very limitation (strict enough, for sometimes), which determines the choice of necessary means of timely receiving of the control information. The smaller this interval, the more strict requests are presented to the software system of continuous planning, considered as a set of mathematical, programming and technical devices of goal purpose. It is assumed, at the same time, that the problem of data gathering is solved satisfactorily. Taking into account the fact, that the decrease of the reaction interval duration causes increasing demands to the information accuracy, one can consider such a form of planning to be an expensive one. It becomes apparent, when it is necessary to apply the high-speed computers, provided with proper peripherals and powerful mathematical software.

Flowing and operative planning problem can be essentially simplified, if one should take into consideration the following properties of the problem flow:

- the problems are regulated in time;
- the real-life problem inputs alter infinitesimally from one problem to another, that is the problems of the flow are "informationally proximal";
- each flow problem can be formulated in terms of a large-scale linear programming problem.

So, for the finding of the optimal solution in a real temporal scale, the package software has to contain:

- flexible means of the informational base transformation;
- some techniques of registration of the optimal solution, obtained on the previous time interval.

Applications of the LP techniques to the solution of prospective planning problems proved its high efficiency. However, to obtain its adequate usage in the case of flowing and operative planning problems, these methods need some additional means enabling evolution of initial data system and the model itself, as well. At the same time the informational evolution and appropriate model state modification must represent the analyzable object dynamics and "have time" to follow it. There are just the iterative techniques from the overall set of LP methods to be applicable to synchronous tracking the analyzable object evolution in the easiest way. This fact is connected with such properties of these techniques as "noise stability", iteration calculative simplicity, program realization invariability relatively to modifications of initial data system. By some model returning, the "expense" of the returning of any computer program of one-iterational realization is negligible.

In spite of the emphasized significance of the iterational techniques in the process of developing object optimization, the finite methods are still really of great importance, and the simplex method - first of all, because of its such properties as exactness, efficiency, possibility to obtain the information total enough to provide economical and mathematical post-optimal analysis results. Being supplied with some means, enabling to begin the calculation with an arbitrary non-basic starting solution, the simplex method can be used side by side with iterative techniques, providing their switching over from one to another on the different phases of the computation. While the package OPTIMA-2 is being designed, the means, securing the problems passage in the characterized mode, have been stipulated, although not to the highest possible degree. The vast positive experience of the solving of economical and

mathematical problems with results, directly applied to the planning practice, was taken into account by the package authors.

2. THE PURPOSE OF THE PACKAGE

The software package OPTIMA-2 is oriented to some concrete problems and methods, i.e.:

- it is a tool, enabling to perform effective testing and instillation of new methods of the LP problems solution to the planning practice;
- the package is oriented to solve various applied problems, which can be formulated in terms of general large-scale LP problems in the mode of flow, regulated in time and satisfying the request of informative "proximity" of the problems, constructing the flow. So, the package OPTIMA-2 problem orientation provides the two-goal purpose. That is, it can be treated as a programming automation tool for experimental researches of the algorithms, at one hand, and as an applied set of computer programs for the real-life problems solution, at the other hand.

3. APPLIED PART OF THE PACKAGE

The package OPTIMA-2 purpose, described above, has determined its applied part.

Today the package contains:

- the simplex method computer program with its necessary service subprograms, intended for the large-scale LP problems solution [5];
- computer programs of iterative procedures, based on the application of methods of penalty functions and revised methods of penalty functions, intended for very the same problems solution [6];
- computer programs of the Fejér-type methods (in their different versions) for the solution of systems of linear inequalities [7].

The revised simplex method with multiplicative representation of the inverse and standard reinverse procedure is used in the OPTIMA-2 package. The mechanism of joining the simplex method and iterational algorithm, enabling to use the available information of the non-basic starting solution, was designed specially for the large-scale LP problems.

The package includes the set of computer programs, making it possible to perform economical and mathematical analysis of the LP problem with consistent constraints, in a rather total way. Such kind of analysis enables to estimate the degree of correspondence of the LP problem formulation to the real-life productional program and to give some specific recommendations on the amending the initial technical and economical data.

By the next package version designation, the abilities to analyse the LP problems, with inconsistent constraints, would be included in it (all theoretical premises to do this are made, and some computer programs, solving inconsistent systems are designed).

4. THE PACKAGE ARCHITECTURE AND SYSTEM PART

The package architecture was mostly influenced by:

- the package supposed users;
- the class of problems, the package is oriented to;
- the utilized software system AVTOKOD-SOMI [8], and the BESM-6 computer operating system (OS) DISPAK [9].

The principal way of the package exploiting by the user is the applying to it with the JOB on the control language, which makes it possible to describe the organization of the problem solution. The each line of the JOB text represents the symbol information of fixed type and simple construction. In the present package version 29 control language operators do functionate. According to their destination, they can be divided into two following types: the dexcription-operators and the action-operators. The first of them give the name to the solving problem, describe the resourses, necessary to solve the problem, give

the initial data and the names to numerical arrays, the package will be working with. The action-operators involve to work the applied modules, which transform the initial numerical data or solve the problem by some technique. Although the language is simple in work, the package is intended to the user, being aware of the LP theory background and techniques. The suitable organization of the LP problem information base is of great importance to the solution. The package OPTIMA-2 provides the data base management system, carrying out two functions:

- the data base creation;
- the interaction with the user.

The data base is created on two magnetic tapes or disks, forming the continuous field. The control language contains special operators, organizing the work with data base and transforming its contents (recording, editing, deletion, condensation, listing).

The structure of information array in the data base is suitable for realization of the LP problems solution techniques and provides the solution in a mode of the flow of "informationally proximal" problems.

The package destination to solve the large-scale LP problems has affected on the module concept in the package. The OPTIMA-2 package module is a computer program itself, according to the definition of the "computer program" concept for the AVTOKOD-SOMI language. All the modules, corresponding to the LP problems algorithms, are got up in this way, and only some of them are formed automatically with assembler texts, i.e. with arbitrary parts of the "computer program". Subsystem ARCHIV(Archives), the component part of the AVTOKOD-SOMI software system, organizes the work with modules archives. It places suitable language means of modules recording, storing, editing and computer programs construction of textual parts at user's disposal [8]. The OPTIMA-2 package archives contains 41 modules, including the system ones, and occupies something about 300 pages on the disk (tape) storage.

So, all the enumerated factors have determined the following composition of the system part of the OPTIMA-2 package:

- the language of jobs, by which the user associates with the package (the control language);
- the package data base;
- the package modules archives, representing applied and system parts of the package ("the package body");
- MONITOR - the main component of the system part, which manages all the other package components.

5. MONITOR OF THE PACKAGE

MONITOR is a multipurpose connection between the BESM-6 computer, OS DISPAK, AVTOKOD-SOMI software system and archives modules, which compose applied and service part of the package. These connections are realized by MONITOR's control services, representing its primary functions:

- control of the package job to work;
- the computer programs control;
- the computer programs interruptions control;
- the information data base control;
- the transit data control;
- control of the package closedown after emergency conditions analysis or after succesful job implementation;
- control of the package work continuation after interruption.

In addition, MONITOR contains a number of service opportunities of gaining statistic data, concerning the package work in a real time. The proper protocol, printed on the alphanumeric printer, informs the user about the problem solution process.

All enumerated control services are founded on the OS DISPAK's basic opportunities, namely, on the apparatus of events and asynchronous processes in the host and subordinate tasks. The apparatus affords to the user OS's opportunities of the host task to handle events, which occure in subordinate task [10, 11]. This apparatus is a complete enough extracode system, ensuring all functions of MONITOR - the host task itself, according to its purpose. Applied and some system modu-

les, controlled by MONITOR, perform subordinate tasks functions, and are connected with MONITOR on the level of extracodes, included in host and subordinate tasks.

The block of event handling in subordinate task is a component part of MONITOR carrying out very important functions of the package work control as a whole. The host task of the present package version reacts on the following events:

- the subordinate task (S.task) has been generated (the S. task has been formed on the input buffer);
- an event has taken place in S. task (S. task has executed extracode "End of task"; S. task has been stopped by the host task; extracode has been intercepted in the S. task, emergency in S. task);
- alarm-clock (the alarm-clock of the S. task event expectation has "given a ring").

In order to synchronize the work of host and subordinate tasks there are set in the host task:

- origin of the interrupt handler program of asynchronous processes and the keeping field origin;
- the events scale mask;
- the alarm-clock.

Control program functions are carried out by MONITOR on the basis of directives given by the user and recorded on the package control language. MONITOR always starts working with forming TRANSLATOR OF JOB - the first subordinate task, which translates the job text into internal problem description, using language tables and the object field model and generating control OPERATORS AND PARAMETERS TABLES for MONITOR and archives modules work. The job syntax and semantics is checked preliminarily (by recording rules control and sense correctness control.)

Computation scheduling is an important function of MONITOR. This problem is solved by the SCHEDULER block - one of the MONITOR's components. Although the sequence of applied modules work in the OPTIMA-2 package is given by the user in JOB, that is the problem of computation scheduling doesn't

arise, the SCHEDULER block generates the modules chain, composed according to JOB, in not the very consecutive order, calling some modules to work by "default". For example, the INFORM computer program, checking the source data for LP problem, works in this mode after working of source data editing modules.

So, SCHEDULER ensures modules control connection - very important MONITOR's function, using for that OPERATORS and PARAMETERS TABLES, created by TRANSLATOR OF JOB, and INTERRUPTION SCALE, initial state of which is ensured by MONITOR. INTERRUPTION SCALE is of great importance to the package work control. It represents one of possible input or output data flows with respect to each subordinate task. Some control messages (e.g., an optimal solution has been obtained, the problem has inconsistent constraints, etc.) are memorized by subordinate task in INTERRUPTION SCALE for their further handling by SCHEDULER.

The GENERATOR block, the second component of MONITOR, composes subordinate task on the input buffer with the help of the OSDISPAK's extracode "Task forming" according to rules of the control cards collecting for the ARCHIVE - subsystem. At the same time the GENERATOR block ensures the application module adjustment to solving problem parameters, using standard AVTOKOD-SOMI (AUTOCODE-SOMI) software system opportunities of assembler texts editing. The GENERATOR computer program is generated also as subordinate task by MONITOR before any applied module forming and calling to work. In order to reduce the hard copy capacity, the operating system hard copy about this task being in a computer has been eliminated by some special OS DISPAK's extracodes. The GENERATOR computer program working is represented in the protocol, printed by MONITOR.

Ensuring of the modules data connection is the next important function of MONITOR. Modules exchange of information arrays is standardized in the OPTIMA-2 package in the following way:

- any module has an access to information arrays of solving problem only through the data-bank;

- if the user needs working with numerical arrays formed outside the package and recorded on the working magnetic tape, then preliminarily given source information must be recorded once again into the data-bank by package facilities;

- MONITOR ensures data-bank magnetic tapes (disks) transmission to each module before it starts working, and takes them away after module close-down.

Special control language operator brings the task in correspondence with data-bank numerical arrays, it will be working with. Names of these arrays are memorized by JOB TRANSLATOR in INTERRUPTION SCALE and are passed together with it to the module, before the last is called to work. INTERRUPTION SCALE is placed on the computer mine storage page No.31₈, which is a transit one with respect to all package modules.

The OPTIMA-2 package is an open-ended system: the user can include an arbitrary procedure in the package at his wish, if the procedure is got up in accordance with AVTOKOD-SOMI software system requirement. In that case he has to name this procedure METHOD and to record it to the package ARCHIVES by this name.

Simplicity of module inclusion in the package is one of its system part virtues. It is necessary for that to include the OSDISPAK's extracode "Waiting the host task to report the solution is possible to be continued" in the module on the level of assembler texts editing. This extracode is put by the user into the very beginning of the module (the first executing command) and into points of "successful" module closedown (more often, instead of extracode "End of task"). All the other module stops on the "End of task", if not replaced by the extracode, named above, are considered to be emergency and are handled by MONITOR especially. So, the new modules inclusion in the package doesn't need considerable expenditure, connected with their alteration.

Now we shall describe MONITOR's computer program management functions. After handling one or several JOB lines, MONITOR

realizes the following operations:

- composes the proper task with the section TELE and with the help of MONITOR's components SCHEDULAR and GENERATOR, using extracode "Forming of the task on the input buffer" and problem description operator for apportionment necessary resources (time and hard copy capacity) to the problem;
- calls the task to the computer channel, using OS DISPAK's extracode "Subordinate task with the section TELE";
- on event "The subordinate task has been generated" the host task establishes for its each subordinate task:
 - a. the keeping field origin;
 - b. the scale mask of events, the host task will be react to;
 - c. the scale of intercepted extracodes.

There are given in the scale mask:

- a. an "abnormal end" in the subordinate task (unit in the 17th bit);
 - b. an extracode has been intercepted (unit in the 19th bit).
- using an extracode "Start-up the subordinate task", MONITOR starts - up the task into translation; while the subordinate task is being translated, MONITOR handles events, ordered in the scale mask (stops, abnormal ends, extracodes); on the translation successful completion, the subordinate task is stopped by the extracode "Waiting the host task to report the solution is possible to be continued";
- on this event execution, MONITOR gives pages on magnetic drum storage, mine storage pages, data-bank working magnetic tapes, the subordinate task must have to work, to it by the use of OS DISPAK's extracodes "Transmission of mine storage pages" and "Transmission of magnetic tape bobbin";
- using extracode "Start - up the subordinate task" once again, MONITOR starts - up the task into calculation and, while computing, handles events, taking place in the subordinate task; module stops by the extracode "Waiting the host task to report the solution is possible to be continued" are considered

to be successful module closedowns; all the other stops in the subordinate task are considered to be abnormal ends and are handled by MONITOR or by OS DISPAK;

- on the module successful closedown MONITOR takes away from the subordinate task all resources, given to it before, and issues and order "Complete the subordinate task";

- after the next JOB line handling, MONITOR generates a new subordinate task and so on, till the control language operator STOP will be executed;

- before the package closedown, MONITOR sends INTERRUPTION SCALE to storage in data bank in case of the package is desired to be used in the mode of computation sequel by the user.

The host task orders an interception only of the extracode of subordinate task call of the array, inserted by the host task, from the input buffer. This extracode is intercepted by MONITOR only in the subordinate task, corresponding to the INPUT control language operator, which is intended for numerical arrays input from cards and for their recording into the data bank. Handling of event "An extracode has been intercepted" is carried on by MONITOR with the help of analysis of some keeping field cells for the subordinate task. As a result of the analysis MONITOR easily recognizes which extracode has been intercepted by that moment. If an extracode, the host task must not react to, has been intercepted, MONITOR issues an order "Continue execution of extracode, intercepted before", and the subordinate task executes the intercepted extracode itself. If a proper extracode has been intercepted, MONITOR chooses an extracode information word from the keeping field and executes it itself. Then MONITOR gives required numerical arrays to the subordinate task with the help of extracode "Transmission of mine storage pages" and issues and order "Start-up the subordinate task".

So, as it follows from what has been said above, almost all MONITOR's functions are reduced to the level of extracodes. The usage of OS DISPAK's basic opportunities has greatly facilitated creation of the OPTIMA-2 package system part. While

adjusting the system part and solving real - life problems, originators of the package satisfied themselves as to the using extracode system sufficient completeness, reliability and efficiency.

Besides enumerated extracodes, while being working MONITOR uses very often:

- extracode "Waiting for events";
- extracode of return to the interrupted place;
- extracode of subordinate task cipher outputs;
- extracode of subordinate task stop causes.

6. PERSPECTIVES OF THE DEVELOPMENT

The OPTIMA-2 package interactive version is being worked out. Interactive mode will afford a number of additional opportunities to the user:

- to give and specify the problem investigation program in the course of connection seance with the package;
- to control the parameters of the computation algorithms, included in the package;
- to respond promptly to different interruptions;
- to be trained in the package working.

This kind of mode is ensured also on the base of OS DISPAK's basic opportunities - special extracode system for the interactive mode organization.

R E F E R E N C E S

- [1] Яненко Н.Н.,
Проблемы математической технологии. - В кн.: Структура и организация пакетов программ: /тез. докл./. Тбилиси, 1976, стр. 9-11.
- [2] Карпов В.Я., Корягин Д.А., Самарский А.А.,
Принципы разработки пакетов прикладных программ для задач математической физики. - Журн. вычисл. математика и мат. физики, 1978, т. 18. №2, стр. 458-467.
- [3] Коновалов А.Н., Яненко Н.Н.,
Модульный принцип построения программ как основа создания пакетов прикладных программ решения задач механики сплошной среды. - В кн.: Комплексы программ мат. физики. Новосибирск, 1972, стр. 48-54.
- [4] Тамм Б.Г., Тыугу Э.Х.,
О создании проблемно-ориентированного обеспечения.- Кибернетика. 1975, №4, стр. 76-85.
- [5] Программы оптимизации. - Свердловск: УНЦ АН СССР, 1974. - Вып. 5. - 140 стр.
- [6] Попов Л.Д., Костина М.А.,
О реализациях итерационных методов для многомерных задач производственного планирования. - В кн.: Методы математического программирования и приложения. Свердловск: УНЦ АН СССР, 1979, стр. 61-65.
- [7] Ерёмин И.И., Мазуров Вл.Д.,
Автоматизация управления параметрами итерационного процесса для задач математического программирования. - Кибернетика, 1979, №6, стр. 41-45.
- [8] Система программирования АВТОКОД-СОМИ для БЭСМ-6. - Свердловск: УНЦ АН СССР, 1974, 111 стр.
- [9] Операционная система ДИСПАК для БЭСМ-6 /И.Д. Бокова, С.А. Зельдинова, В.И. Зуев и др. - М., 1976, 80 стр./- Препринт/ИПМ им. М.В. Келдыша; №1/.
- [10] Зельдинова С.А., Паремский М.В., Тюрин В.Ф.,
Некоторые базовые возможности ОС ДИСПАК.- М., 1976, 80 стр. /Препринт/ИПМ им. М.В.Келдыша/.
- [11] Тюрин В.Ф.,
События и асинхронные процессы в главной и подчиненных задачах в ОС ДИСПАК.- М., 1980, 20 стр., /Препринт/ИПМ им. М.В. Келдыша; №13/.

DIALOGUE SYSTEM FOR OPTIMIZATION

Yu. Evtushenko, E. Veselov, V. Mazurik

(Moscow, USSR)

INTRODUCTION

Since 1977 a research on the dialogue system for optimization DISO has been carried out in the Computer Center of the USSR Academy of Sciences. DISO is intended for unconstrained and nonlinear optimization and optimal control problem solving. With the beginning of the Working Group-7's activity the development of the DISO was coordinated with the WG-7 plans. Participation in the WG-7 works proved to be very useful for new optimization method development and their testing on the tasks which the WG-7 members exchanged. In particular, in the WG-7 activity scope the Computer Center cooperates with the Technical University of Dresden, GDR, in the field of optimization methods exchange, their testing and control program architecture unification.

The DISO is used for application problem solving since 1979. During this time great experience has been accumulated on the library content, architecture decisions effectivity, input language suitability etc.

The experience of DISO development can prove useful for the other WG-7 members, because dialogue systems architecture due to the same problem scope has many features in common for different optimization systems.

As a rule, the user is not entirely satisfied with the system at his disposal. The availability of the adaptation capabilities in the system is the basic condition of its effective exploitation. Such capabilities has been developed in the DISO system. The concepts used are of general character and

therefore they can be the base adaptation capabilities in some other dialogue packages.

1. DISO ARCHITECTURE

DISO is intended for solving complex practical optimization problems in dialogue.

If we need a system which is not only a scientific experiment or unique laboratory set, but effective product for mass usage, then we need not restrict the system to the set of numerical methods. The system should also include effective software of all technological operations, which constitute the actual procedure of a given task.

Every task can be defined as a set of certain components - functional relations, data sets, operations, subtasks etc. Technological operations like storing program components in files, editing and loading them should account for mutual links and component dependence due to the specific of problem orientation. It can be realized only when such technological operations are performed on a problem-oriented level.

For example DISO editor should take into account, that the program units are not independent, but are combined in such ansambles as "restriction set, which defines feasible space", or "differential equation set which defines the phase trajectory" etc. Such program units concern a certain task and therefore a change of, say, an independent variable dimension should result in the change of the corresponding data sets. Such a problem-oriented relation should influence the file system organization and dialogue monitor input language.

Different problem orientations need new technological subsystems having analogues possibilities. It is not worth while to redesign such systems again. This means, that they should be constructed on a universal instrumental basis following the principle of extensibility.

This is just a way, DISO is organized. It has all the components, necessary for problem-oriented technology of task solving. Such DISO subsystems as text editor, dialogue monitor, file system are realized as problem-oriented level on the more universal level of base subsystems. For the other problem orientations only this upper level must be changed. The unification of all basic systems program realization in DISO is achieved by the use of instrumental system called DIALOGUE intended for implementation dialogue packages like DISO.

From the user's point of view DISO has the structure, representing on Fig.1.

2. BASE SUBSYSTEM

Base subsystem containing the numerical method library and interactive monitor is the main component of DISO.

2.1 Numerical method library

DISO numerical method library includes several sets of methods corresponding to different types of optimization problems: unconstrained minimization, nonlinear programming, optimal control. Optimization procedures of these three classes underlie a wide variety of numerical mathematic problems which makes DISO a universal tool for their solution.

The availability of methods intended for different optimization task characteristics was the criteria of the library completeness. The characteristics in question are point feasibility remoteness from the solution point, function smoothness etc. Effective solution of the optimization problems is achieved by the combined application of different methods.

The following is the library content.

Unconstrained minimization methods:

- coordinate search,
- steepest descent,

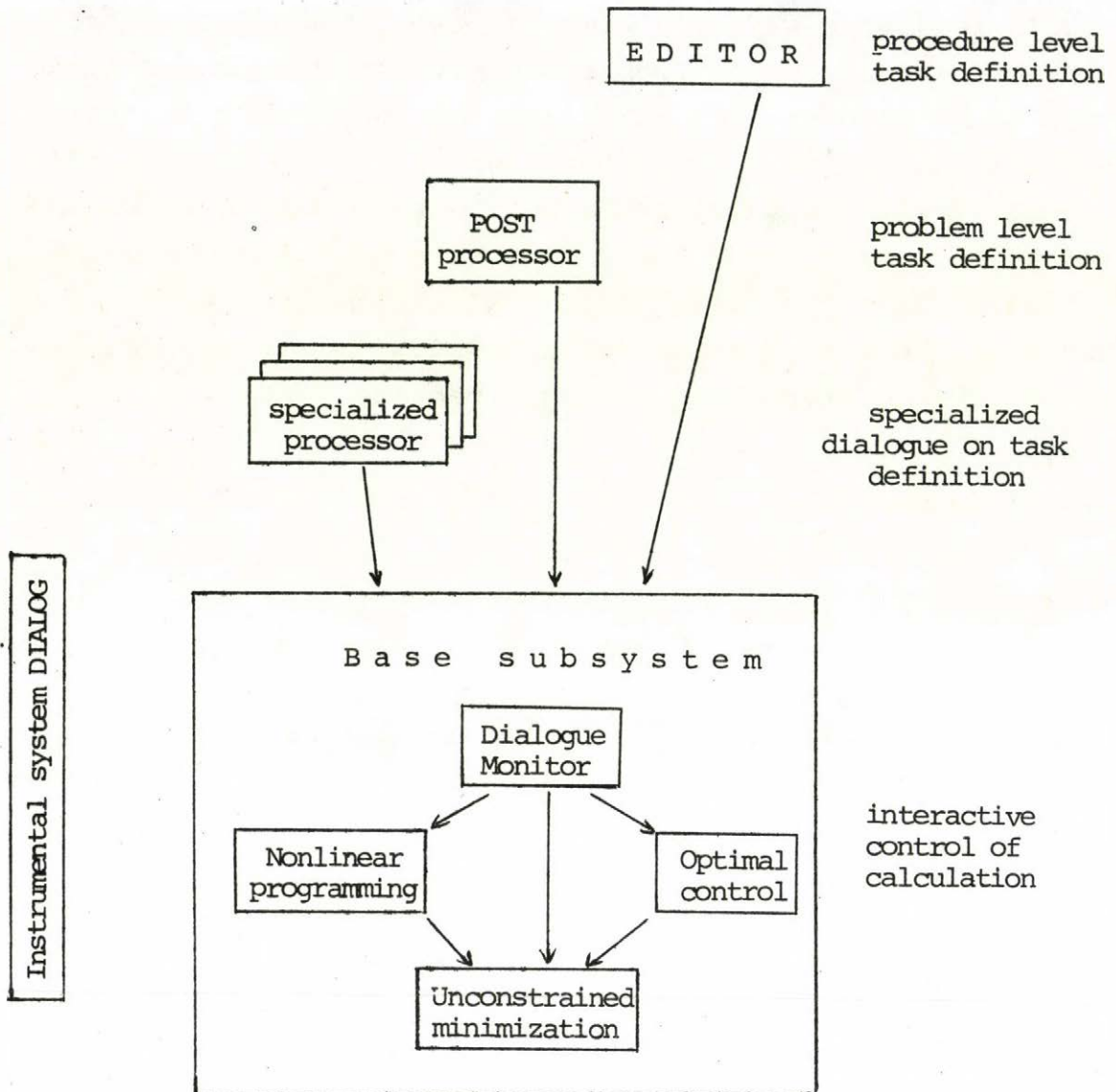


Figure 1. DISO architecture

- conjugate gradient,
- Hooke and Jeeves
- conjugate-direction method of Powell,
- random search,
- modified Newton,
- quazi-Newton,
- ellipsoidal method,
- variable metric method (Broyden, Fletcher, Shanno).

Nonlinear programming methods:

- feasible direction,
- linearization,
- primal method,
- penalty function methods,
- augmented Lagrangian method,
- generalized Lagrangian methods,
- modified relaxation,
- modified Newton method,
- modified quazi-Newton method,
- Morrison method,
- modified Morrison method.

Optimal control methods in DISO library permit to solve problems with path and terminal constraints. Most algorithms are based on finite-dimensional representation of optimal control problems as nonlinear problem which is solved mainly by a standard NLP algorithms. To treat the OCP by nonlinear programming it is necessary to use a finite number of parameters to define the control function. For this purpose we divide the time interval into subintervals and use a finite set of parameters to represent the control in each subinterval. The system of problem equations is integrated using Euler, modified Euler and Runge-Kutta formulae. For all these cases the formulae for computation of first and second derivatives of performance functional with respect to the decision variables and design parameters were obtained. These results permit the use of a wide class of first and second order NLP methods. Several numerical methods were based on utilization of discrete Pontryagin maximum principle.

Optimal control methods:

- penalty function,
- augmented Lagrangian,
- generalized Lagrangian,
- Morrison method,
- modified Morrison method,
- linearization method,
- feasible direction method.

2.2 Dialogue monitor

DISO dialogue monitor provides flexible interaction of a user with the optimal problem solving process. The monitor gives the opportunity to change optimization method in the calculation, to analyse and correct the current state variables on all the levels of the computational process, to adapt method control parameters, to calculate the values of functional objects (goal function, constraints, etc) to use some service operations etc.

Dialogue monitor capabilities take into account not only needs of mathematical character, but the technological ones. Optimization process control is fulfilled not only in a dialogue, but in automatic manner using a specialized task control language which in main coincides with interactive control language.

For the most DISO problem types computational process is multilevel. Therefore process control can be carried out on several levels. For example, there can be unconstrained minimization subtask for some nonlinear methods. The dialogue and automatic (by scenario, performed in advance) regimes can be flexibly distributed over such hierarchy levels.

Highly developed informational system intended for in-line user teaching is at user's disposal.

All the enumerated mathematical and technological capabilities are implemented with the help of DISO instrumental sys-

tem and therefore they are not entirely unchangeable. The user can adapt them for a certain specific area.

3. TASK DEFINITION CAPABILITIES

As important as solving process control step is a task definition step. The proper problem-oriented capabilities were developed in DISO for this step of problem solving.

3.1 Problem-oriented task definition

The main aim of this level is a model synthesis and its analysis. The emphasis in the dialogue is laid on the description and analysis of objects and relations, constituting the task definition.

High level language POST is DISO part which implements problem oriented level.

The POST language allows to describe models as ansambles of objects and their relationships. Object types of applied mathematics, such as scalar, vectors and matrices are possible. Object relationships are defined algorithmically as ALGOL-like procedures. POST includes scalar and matrix arithmetic and structured control constructions of selection and iteration. The basic feature of the language is the availability of task definition statements. The language includes a number of pre-defined statements for optimization, algebraic equation system solution, ordinary differential equation system integration. The extension of this set by user-defined task statements is allowed. Executing the task definition statement POST interpreter picks up the corresponding objects and procedures from model and transfer then to special dialogue monitor, corresponding to a given type of task.

As it was mentioned above optimization procedures are the foundation for a great number of applied mathematics algorithms. The concept of differential is the essential concept of applied

mathematics. The ideological basis of POST language is the inclusion of differential notion into the language semantics. The implicit differentiation mechanism is implemented. At user's disposal are syntactical constructions with the help of which he can switch the differentiation process at any point of the program. As a result he is allowed an access not only to the values but to their derivatives. At user's indication either the first or both the first and the second derivatives are calculated. The nested differentiation construction are allowed resulting the possibility of partial differentiation.

The peculiarity of implemented differentiation mechanism is the fact that the exact derivative values, coinciding with analytical values are calculated.

In a dialogue session the POST interpretator executes the user's commands regarding them as operators of a certain global procedure which was initiated at the beginning of the session and will be over at the moment of its completion. The task definition and differentiation statements are competent POST statements and so they can be included into the procedure body. Therefore in the dialogue session the user can invoke any model procedure, formulate and solve tasks, based on the model information, control the differentiation processes. If he invokes the procedure, which includes the task definition statement, the initiation of a dialogue, corresponding to this task will take place. In this way a rather complicated hierarchy dialogue structure can be realized.

3.2 Procedure-oriented task definition

The above problem-oriented means are intended for a case when task definition is not fixed a priori, but must be formed or modified in dialogue. The dialogue goal in this case is to find the adequate and practically solvable task definition.

Frequently the situations of another kind occur as well, when optimization is only one step of a certain general cal-

culational procedure. In this case the optimization task definition is fixed and need not to be modified. Therefore another capabilities of task definition are used. The most suitable way is to invoke some special procedure supplying it with task definition through the parameter. The invocation is realised from a certain general calculational program written in one of the traditional languages like FORTRAN.

In this case the ordinary text editor is needed for task definition because this definition is represented as some set of procedures on programming languages. Dialogue monitor for procedure linking and executing is also needed. The universal text editor and dialogue monitor are components of DISO which maintain the procedure-oriented interface level.

On this level the use of DISO mathematical capabilities is realised by direct call of base subsystem control procedures. Each class of optimization tasks has its own control procedure in the base subsystem. The invocation of such a procedure allows the dialogue solution of the task. Note that the language POST mentioned above provides the optimization task solution with the help of the same base subsystem control procedures, but besides it also provides the problem-oriented verification of task structure and automatic creation of effective differentiation procedures.

4. INSTRUMENTAL CAPABILITIES

As it was mentioned above the basic idea of DISO implementation is the use of a common instrumental base in order to achieve real extentionability. The instrumental system DIALOG incorporated in DISO provides this very base.

The problem-oriented interface level can be subdivided into two sublevels: an application and a model one. The application-oriented level is intended for utilizing it by applied user with his own terminology and task specification.

On the contrary the model-oriented level is suitable for creation and development of the terminal application-oriented systems. This level is most adequate for the description of notions and relations, inherent to the applied mathematics models. The procedure-oriented level, on which numerical algorithms are implemented underlies the model-oriented level. This level classification is convenient for characterization of DIALOG system status in DISO.

The role of DIALOG is support model-oriented technology. The DIALOG input language allows to describe the information structure of mathematical models and operational structure of their correlation in dialogue.

Both structures are implemented on relational and algebraic basis. On the program level they are based on service procedures for exceptions execution, data base management, lexical and sintactical text analysis, storage control etc. All procedures of this kind are necessary in computer implementing of any mathematical model, but from the point of view of applied user they are alien. The fact they are concealed from the user by linguistic structures of relational and algebraic character essentially facilitates the implementation of such models.

Kodd's relational basis is used for informational structure description. Axiomatic method is used for automatic management of structure relationships in model.

As a theoretical base for dialogue structure description and analysis Petri nets were chosen. It allows the unified description of dialogue language, parallelism of dialogue operations, the connection of dialogue structure with the relational information structure of model.

Relational and algebraic base of DISO seems to be rather perspective for two reasons. The first, it is full in the sense of expressive power. The second, it is quite technological in the sense of reliability and effectivity: relational structures allow to maintain the model integrity with ease when a great number of model interrelated components is available, simultaneous usage of several model variants and versions takes place etc.

5. CONCLUSION

At present the DISO system is implemented within a framework of monitor system on the computer.

The minimum configuration of DISO includes the basic subsystem, which provides the procedure-oriented level facilities. The user can either use his own text editor and dialogue monitor, or take them as an addition to base subsystem. To work with DISO at a problem-oriented level the subsystem POST must be included into the above set.

In addition, the other problem-oriented subsystems of both application and model levels can be included in DISO by the user himself with the help of the instrumental system DIALOG. This allows to adapt DISO capabilities to user's specific needs.

The research is being made now to transfer the DISO system to the computers of EC and CM (PDP) series.

PROCEDURES FOR OPTIMIZING ECONOMIC MODELS

U.H.Malkov, V.A.Skokov, B.V.Cherkasskiy

(Moscow, USSR)

Packages of procedures for optimizing economic models (POEM) was implemented at the Central Economic and Mathematical Research Institute of the Academy of Sciences of the USSR for the ES Series of computers.

The package was implemented for solving linear, quadratic and nonlinear programs as well as transportation ones. It includes wide range of algorithms. The package is an open systematized library of procedures coded in PL/1 and FORTRAN.

The existing package PMP [7], LP ASU [17] and PPP TP [18] for ES EVM do not deal with sufficiently wide class of practical problems: especially concerning maximal flow, transport and nonlinear problems. Also it is complicated to use existing packages as procedures investigating sophisticated problems that include optimizing models (as parts of them). Therefore package POEM was elaborated as systematized library of FORTRAN and PL/1 procedures using typical mathematical software of ES EVM. The algorithms implemented in the package POEM are based on recent result in theory and practice of mathematical programming, that have been published in various journals, proceedings and other editions. The high efficiency of these algorithms is supported by practical solving of problems.

Package POEM consists of the following three sections:

- modules for linear and quadratic programs
- modules for nonlinear programs
- modules for solving transport problems (matrix, network and maximal flow).

The linear programming modules are elaborated to solve

$$\max \left\{ \sum_{j=1}^n c_j x_j \mid \sum_{j=1}^n a_{ij} x_j = b_i \ (i=1, \dots, m), \ 0 \leq x_j \leq \hat{x}_j \ (j=1, \dots, n) \right\}$$

The package includes modules that realise direct algorithm of simplex method, some variants of multiplicative algorithm and an iterative algorithm that uses the modified Lagrangian with penalty vector. Modules were intended for solving problems of mid dimension. If the user has got 90 K bytes of CORE at his disposal for storing the code he uses and both initial and intermediate data, then it is possible to solve problems up to $m \cdot n = 200 \cdot 400$ and the number of nonzero elements up to 1500. If the CORE resources are 150 K then it is possible to solve problems with up to 500 constraints.

In order to solve small problems (up to 100 constraints) where the number of nonzero elements in the matrix is comparatively high (30-50%) it is preferable to use the direct algorithm of the simplex method SMPLEX.

It is well known that the most effective algorithm for solving sparse linear programs (with sparse coefficient matrices) is of course a multiplicative algorithm of the simplex method [11]. Offered variants of the multiplicative algorithm (coded in PL/1 and FORTRAN, with either take into account of the upper and lower bounds of the variables or not) - MAPLB1, MAFTB0, MAFTDN - realize the compact form of the algorithm. Combined storing of initial matrix and multiplicative representation of triangular factorization of basic inverse is the main feature of the algorithm.

Only multipliers (i.e. eta-vectors) corresponding to spikes - columns which do not allow to transform the basic matrix to triangular form by row and column permutations - are stored explicitly after the reinversion. In the INVERT operation ideas similar to those of Hellerman and Rarick [36] are used. Such method of data storing permits to enlarge the size of the problems to be solved.

For user's convenience and to facilitate solving practical LP problems by the computer ES EVM the package includes codes FROMPS and LINPUT for conversion of initial data from the MPS' format into the format of modules of the package POEM and vice versa. MPS' format is less compact. The package also includes an iterative code ITLIMF which realizes new and effective algorithm, similar to the multiplier method [5] and to the method of Bertsekas [37].

The implemented iterative algorithm is a dual type algorithm based on using the modified Lagrangian

$$F(x, y, \alpha) = (c, x) + (y, b - Ax) - \sum_{i=1}^m \frac{\alpha_i}{2} (b - Ax)_i^2$$

The main steps of the algorithm on the (S+1)-th iteration are the following:

Step 1. Approximate solving the auxiliary problem by the method of coordinate descend

$$x^{S+1} \approx \arg \max_{x \geq 0} F(x, y^S, \alpha^S).$$

Step 2. Recalculation of dual solution (of gradient type):

$$y_i^{S+1} = y_i^S - \alpha_i^S (b - Ax^{S+1})_i.$$

Step 3. Recalculation of penalty vector.

$$\alpha_i^{S+1} = \Phi_i^S(x^S, x^{S+1}, y^{S+1}, \alpha^S).$$

The main difference between the described algorithm and the former iterative LP procedures of this type is expressed by transit from penalty scalar coefficient to penalty vector α , which makes it possible to take into account special features of each constraint of the problem we are solving.

The described iterative algorithm enables us to get the solution of LP problems also with the accuracy of 1%. In this

case the number of iterations is comparable with the number of iterations of the simplex method.

In order to solve quadratic programming problems the package includes 2 modules. The first one - QPSUB 1 realizes iterative algorithm based on the method of penalty estimations [21, 2] where instead of initial problems the following one is solved

$$\max_y \min_{\{v \leq x \leq w\}} \{c^T x + x^T Q x + (y, Ax - b) + \alpha \|Ax - b\|^2\},$$

Calculation scheme for minimizing x realizes the method of conjugate gradients with taking into account the bounds on the variables. The solution time is closely connected with proper choice of initial point and calculation control parameters. The tighter bounds on the variables are given the more quickly the process converges.

The second module of quadratic programming is named QPWOLF. It implements the modified variant of Wolfe's method [9] where on the base of Kuhn-Tucker's conditions a linear problem

$$Ax = b; -2Qx + A^T \lambda + v = c; x, v \geq 0,$$

under the nonlinear constraints

$$x_j v_j = 0 \quad (j=1, \dots, n)$$

and with free variables λ is solved instead of the initial one.

In order to solve the described problem a multiplicative algorithm is used that takes into account the presence of mentioned nonlinear constraints and free variables.

There is a great variety in formulation of nonlinear programming problems. Any class of nonlinear programming problems has certain form of initial data. Therefore in the POEM

nonlinear programming section the division of subroutines into problem oriented complexes is adopted. Nonlinear programming subroutines can be used for solving problems with differentiable or nondifferentiable functions. The POEM unconstrained minimization section consists of the following complexes: the complex of subroutines for minimization of differentiable functions with calculation of derivatives and without it, the complex of subroutines for minimization of nondifferentiable functions and the complex for minimization the sum of quadratic functions. The POEM constrained minimization section includes complex of subroutines for minimization of differentiable function subject to lower and upper bounds, complex of subroutines dealing with geometric programming and complex of subroutines dealing with general problem of nonlinear programming. The main features of nonlinear programming section are the realization of contemporary fast operating techniques, optimal use of storage (using of compact form of information), reliability of subroutines (these subroutines obtain a solution for sufficiently large set of problems), modular structure of all subroutines. All subroutines are written in FORTRAN-IV and have two variations (calculation with ordinary and double precision).

The subroutines of complex with identical form of initial data have identical form of call. This makes these subroutines interchangeable. It is only the title of subroutines which is to be changed when the other subroutine is used. All subroutines are supplied by a subroutine, printing the intermediate results with controlled frequency of listing information and its volume. Any interruption of calculation is accompanied by diagnostics of interruption cause and there is a possibility to establish different regimes to continue calculations.

There is some mnemonics in the titles of the nonlinear programming subroutines. Particularly, the last letter of the title points to the precision of calculation. If the subroutine operates with ordinary precision, the last letter of the title is R, and in the other case - D. Briefly, we shall mention only title concerning double precision.

For solving unconstrained minimization problem

$$\min\{f(x) | x \in E^n\}$$

there are three complexes in POEM. In the complex using derivatives (subroutines MU110D, MU120D, MU130D, MU140D, MU160D, MU181D) the following methods for differentiable functions are realized: conjugate gradient method by Fletcher-Rieves, conjugate gradient method by Poljak-Polak-Ribiere, quasi-Newton method by Davidon-Fletcher-Powell, quasi-Newton methods by Broyden and quasi-Newton method by Broyden-Fletcher-Shanno [21,27]. A code-generator of testing functions was used for testing these and other methods. This code generates different functions with determined differential and topological properties where dimension and condition number may be varied.

The complex of subroutines MU010D, MU020D, MU030D, MU040D, MU060D includes the subroutines of previous complex with difference approximation of gradient and the subroutine which realize the conjugate direction method by Powell. There are two types of errors when we calculate the difference approximation of gradient by computer: the truncation and the cancellation errors. The scheme which is realized in these subroutines provides the optimal choice of approximation steps which guarantees minimum calculation error and retain the rate of convergence of conjugate direction methods [13]. All subroutines of both complexes use cubic interpolation line search. The complex of subroutines which minimize nondifferentiable functions contains subroutines MUD01D, MUD03D in which ellipsoid of two consequent subgradients are realized [12,29,24].

In the POEM there are two codes (MS121D, MS12KD) for function minimization in the sum of quadratic form. These subroutines realize modification of Gauss-Newton method [19]. Moreover, the second one can be used for example to solve a system of nonlinear equations with sparse Jacobian. This code uses compact form to store Jacobian. When the quadratic approximation problems are solved we may obtain and list some

results concerning statistical analysis of final approximation, experimental and calculated curve.

Complex of subroutines MC120D, MC130D deals with the following problem:

$$\min\{f(x) \mid x \in Q\}, Q = \{x \in E^n \mid l \leq x \leq u\},$$

in these subroutines some variants of reduced gradient method with controlled sequence are realized [23]. These variants are based on conditional gradient method by Poljak-Polak and quasi-Newton method by Fletcher [27,15]. Complex of subroutines GP120D, GP130D deals with geometric programming problem

$$\min\{f_0(x) \mid f_i(x) \leq 1, i=1, \dots, k; x > 0\},$$

where f_i is posinom [26], i.e. generalized polinom with positive coefficients.

To determine and solve this problem only values of coefficients and matrix of exponents (orders of polinoms) are needed.

In subroutines the method of penalty estimations for solving dual problem of geometrical programming is realized. The great advantage of these subroutines over another code (for example [38]) consists in application of compact form to store matrix of exponents.

For solving nonlinear programming problems with general constraints

$$\min\{f(x) \mid g_i(x) = 0, i=1, \dots, m \quad l \leq x \leq u\},$$

where f and g_i are enough smooth functions there is a complex of subroutines AC120D, AC130D in POEM. They realized method of penalty estimations with different techniques to solve auxiliary problems. Numerical examination of this method [22,25] showed its high efficiency. First of all these subroutines deal with problems which have sparse Jacobian of boundary function. These subroutines use compact form to store Jacobian.

In the section of the package that is devoted to network problems, there are presented procedures meant for solving linear transportation problems in various forms, procedures for finding shortest paths in network, for constructing maximal network flow and for solving three-indexed transportation problem with planar sums.

Cited problems can be regarded as classicals. Codes for solving such kind of problems are often needed in complexes of programs where they are used as a rule not only once but as auxiliary optimizing procedures when executing some iterative algorithm. On authors opinion, programmers who are including such codes into their own do not need any service on high level. What they expect from codes is following: reliability, maximal efficiency, a simplicity of using. We hope that procedures gathered here satisfy the formulated requirements. Algorithms realized in the elaborated procedures are regarded as the best in cited literature.

Procedures gathered in the package include sufficiently wide class of linear transportation models that enables the user to choose more suitable procedure for solving his problem.

Procedures presented in the package realise primal method using list structures for storing basis information [8,3]. Using such structures enables to achieve top efficiency during one iteration, that is expressed by the fact that recalculation is needed for those and only for those elements of the base that really are going to be changed on that iteration. Codes that use analogous list structures were independently elaborated in the USA [34,30,35] and are the best for solving transportation problems at the present time.

For problems that deal with finding shortest paths in network a lot of effective algorithms is known. One of those is an algorithm of two-side queue [10,39]. Another approach is a good realization of Dijkstra algorithm [31,1,6,32]. In the codes presented in the package both approaches are realized.

Procedures for sorting arcs in the network have auxiliary character. They are necessary for preparing network before finding shortest paths and can be used for shortening solution time of transshipment problems.

Problem of constructing maximal flow can be considered as a special case of a transshipment problem.

For solving this problem in the majority of cases it is however expedient to use special methods that have proved their reliability in vast experiment presented in [33]. The package includes a procedure that realize special algorithm described in [28]. By the information presented in this paper this algorithm is the best.

Procedure for solving three-indexed transportation problem with planar sums in matrix formulation belongs to the class of codes that realize iterative methods. It is based on a variant of a penalty method described in [4]. Authors have not found any references about codes that really solve large-scale three-indexed transportation problems. The proposed procedure is apparently first step in this direction.

We give here the list of procedures in this section of the package:

NCONTR, NTRANS, NEXTR (PL/1) - procedures for solving uncapacitated transshipment problems.

NCONTR - is meant for preliminary process of information,

NTRANS - for input information and output results,

NEXTR - for construction of optimal solution,

MCONTR, MTRANS, MEXTR (PL/1) - procedures for solving uncapacitated transportation problems,

CNCONTR, CNTRANS, CNEXTR (PL/1) - procedures for solving capacitated transshipment problems,

CMCONTR, CMTRANS, CMEXTR (PL/1) - procedures for solving capacitated transportation problems,

NEXTRF, MEXTRF (FORTRAN) - procedures for construction optimal solution for according uncapacitated transshipment and transportation problems.

IPU, KRAPUT (PL/1), IPUF, KRAPUF (FORTRAN) -
procedures for finding shortest paths in the network,
PEREST, PERESTC (PL/1), PERESF (FORTRAN) -
procedures for sorting paths in the network,
CHFL1 (PL/1) - procedure for constructing maximal network flow,
MATR1 (PL/1) - procedure for iterative solution of three-in-
dexed transportation problem with planar sums in matrix for-
mulation.

R E F E R E N C E S

- [1] Anikeich A.A., Gribov A.B.
The algorithm of queues for finding shortest paths in directed graph (in Russian).
In "Upravlenie transportnimi processami", Moscow, Nauka, 1977.
- [2] Belov E.N.
An algorithm of the conjugate gradient and multiplier method for solving linear and quadratic programming problems.
Coll. Codes and algorithms, N 66, CEMI Ac. of Sciences, M., 1976.
- [3] Bronstein M.L., Gohvat E.L., Kim K.V., Cherkassky B.V.
An efficient algorithm for transformation the basis for transshipment problem (in Russian).
Proceedings of IKTP, Moscow, v.67, 1977, 168-181.
- [4] Vereskov A.I.
A variant of penalty algorithm for problems with row structure of blocks (in Russian).
Economica i Matemat.Metodi, v.14, n.5, 1978.
- [5] Golstein E.G., Tretiakov N.V.
The gradient minimization method and convex programming algorithms, concerned with modified Lagrangian. (in Russian).
Economica i matemat. metodi, 1975, v.XI, N.4.
- [6] Dinits E.A.
Economic algorithms for finding shortest paths in network. (in Russian).
Proceedings of VNIISI, Moscow, 1978, n.4, 36-44.
- [7] Catalogue of the ES EVM, I.2. Software.

- [8] Kim K.V.
About the efficiency of algorithms for solving
bicomponent problems of linear programming. (in Russian)
Económica i matemat. metodi, 1974, v.10, N.3, 621-631.
- [9] Künzi H.P., Krelle W.
Nichtlineare Programmierung. (in German)
Springer-Verlag, Berlin, Göttingen, Heidelberg, 1962.
- [10] Levit B.Yu., Livshits V.N.
Nonlinear transshipment problems. (in Russian).
Moscow, Transport, 1972.
- [11] Malkov U.H.
Efficiency improving technique for the product form
inverse algorithm of the simplex method. Review.
(in Russian).
Coll.N.7. Mathematical methods to solve economic
problems. Nauka, 1977. Moscow.
- [12] Nemirovskii A.S., Judin D.B.
Problem complexity and the efficiency of the optimization
methods. (in Russian).
Moscow, Nauka, 1979.
- [13] Nesterov Ju.E.
A scheme of gradient approximation in conjugate direc-
tions methods. (in Russian).
Coll. "Comput. Meth. of Math. Program". CEMI Ac. of Sc.
M, 1980.
- [14] Nesterov Ju.E., Skokov V.A.
Testing unconstrained minimization algorithms. (in
Russian).
Coll. "Comput. Meth. of Math. Program". CEMI Ac. of Sc.
M, 1980.

- [15] Nesterov Ju.E. Skokov V.A.
First order methods of nonlinear unconstrained minimization. (in Russian).
Coll. "Comput. Meth. of Math. Program". CEMI Ac. of Sc., M., 1980.
- [16] The package of the procedures for optimizing economic models. ("POEM"). (in Russian).
General description. CEMI Ac. of Sciences, M., 1980.
- [17] The package "Linear programming in ASU". (in Russian).
Application description. Kalinin, NPO "Centrprogram-system", 1978.
- [18] The package "Transportation problem". (in Russian).
The application description. Kalinin, NPO "Centrprogram-system", 1978.
- [19] Polyak B.T., Skokov V.A.
Solving of problems on minimization of sum squares.
(in Russian)
Economika i Matemat.Metodi, v.14, iss.6, 1973.
- [20] Polyak B.T., Tretiakov N.V.
Multiplier method for problems on constrained extremum.
(in Russian)
I. of Math. and Math. Physics, 1973, v.13, N.1.
- [21] Polyak B.T., Tretiakov N.V.
An iterative method of linear programming and its economical interpretation. (in Russian).
Ec. and Math. Met., 1972, v. VIII, N.5.
- [22] Skokov V.A.
Algorithm of multipliers for minimization of several variables function with constraints (ALGOL). (in Russian)
Codes and algorithms. CEMI Ac. of Sc., v.55, M., 1973.
- [23] Skokov V.A.
Algorithm for solving problem of minimization of several variables function with two-bounds constraints. (in Russian).
Coll."Codes and Algorithms", CEMI Ac.of Sc., v.75,M., 1977.

- [24] Skokov V.A.
Some remarks to minimization methods concerned with space scaling. (in Russian).
Cybernetics, 1974. N.4.
- [25] Skokov V.A.
Computational experience in solving of nonlinear programming problems. (in Russian).
Math.Methods of solving of ec. problems. Coll.N.7, M., Nauka, 1977.
- [26] Skokov V.A., Gazieva A.A.
An algorithm for solving geometrical programming problems. Coll. "Programs and algorithms", N.78, CEMI Ac.of Sc., M., 1977.
- [27] Himmelblau D.
Applied nonlinear programming. (in Russian).
M., Mir, 1975.
- [28] Cherkassky B.V.
Procedures for constructing maximal network flow. (in Russian).
Proceedings of VNIISI, M., 1979,n.3, 90-96.
- [29] Shor N.Z.
Minimization methods for nondifferentiable functions and applications.
Kiev, Naukova Dumka, 1979.
- [30] G.Bradley, G.Brown, G.Graves.
Design and Implementation of Large-Scale Primal Transshipment Algorithms.Man.Sci., v.24, n:1, 1977,1-34.
- [31] E.W.Dijkstra
A note on two problems in connections with graphs.
"Numerical Mathematics", v.1,1959, 269-271.
- [32] J.Gilsinn, C.Witzgal
A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees.
NBS Tech. Note 772, US Dep. of Commerce, 1973.

- [33] F.Glover, D.Klingman, J.Mote, D.Whitman
Comprehensive Computer Evaluation and Enhancement of
Maximum Flow Algorithms.
Res. Rep. CCS 356, Center Cyb. Studies, u. of Texas at
Austin, 1979.
- [34] F.Glover, D.Karney, D.Klingman
Implementation and comparisons of Primal, Dual and
Primal-Dual Computer Codes for minimum Cost Network Flow
Problems.
Networks, v.4, 1974, 191-212.
- [35] M.Grioriadis, T.Hsu.
The Rutger Minimum Cost Network Flow Subroutine
Sigmap, 26 (1979), 17-18.
- [36] Hellerman E., Rarick D.
Reinversion with preassigned pivot procedure
Math. Progr., 1972, v.2, 263-278.
- [37] Kort B.W., Bertsekas D.P.
Combined Primal-Dual and Penalty methods for Convex
programming.
SIAM J. Control and Optimization, v.14, N.2. 1976.
- [38] J.Z.Kuester, I.H.Mize
Optimization techniques with Fortran
Mc. Graw-Hill Book Company, 1973.
- [39] U.Pape
Implementation and Efficiency of Moore-Algorithms for
the Shortest Route Problem.
Math.Prog., 7 (1974), 212-222.
- [40] Shanno M.I.D.
Conjugate gradient method with inexact search.
Mathematics of Operation Research.
1978. v.3, N.3.

THE PACKAGE OF APPLIED PROGRAMS "KVAZAR" FOR ANALYSIS OF INCONSISTENT CONSTRAINTS THROUGH COMMITTEE CONSTRUCTIONS

VI. Mazurov, V.S. Kazantzev

(Sverdlovsk, USSR)

1. INTRODUCTION

For analysis of system of constraints

$$f_j(x) \leq 0 \quad (j=1, \dots, m), \quad x \in R^n, \quad (1)$$

which is not assumed to be consistent, we use the following constructions: maximal (in sense of inclusion) consistent subsystem of (1); minimal (in sense of inclusion) inconsistent subsystem; and committee.

Definition. The committee of system (1) is called the set $K = \{x^1, \dots, x^q\} \subset R^n$ such that for each $j \in \overline{1, m}$ the majority of inequalities

$$f_j(x^i) \leq 0 \quad (i=1, \dots, q)$$

are satisfied.

These conceptions are applied to research of following two aspects of problem

$$\sup\{f(x) : x \in M\}, \quad M = \{x : (1)\} : \quad (2)$$

- identification of poorly formalized restrictions;
- finding of optimal vector.

2. IDENTIFICATION OF POORLY FORMALIZED RESTRICTIONS

Let one restriction

$$f_1(x) \leq 0 \quad (3)$$

of system (1) be unknown. This restriction may be poorly formalized, that is the information about them is not given in analytical form but the finite sets are given of examples of feasible and not feasible vectors according to this restriction. Thus, we assume, that the finite sets are known: $A \subset R^n$, $B \subset R^n$, and

$$f_1(x) \leq 0 (\forall x \in A), \quad f_1(x) > 0 (\forall x \in B).$$

Then we may attempt to find the model of restriction (3) through solving of discriminant analysis problem for sets A and B . This is the following problem. When the class Φ of functions is given, we find

$$\varphi \in \Phi, \quad \varphi(x) \leq 0 (\forall x \in A), \quad \varphi(x) > 0 (\forall x \in B) \quad (4)$$

The inequality $\varphi(x) \leq 0$ is model of restriction (3).

The problem (4) is one of the pattern recognition problems. If the class Φ is too small, this problem may be inconsistent (insolvable). In this case we may attempt to find the committee

$$K' = \{\varphi_1, \dots, \varphi_q\} \subset \Phi$$

that is the set K' , which satisfies the conditions: for each $x \in A$ are right more than half of inequalities

$$\varphi_1(x) \leq 0, \dots, \varphi_q(x) \leq 0$$

and for each $x \in B$ are right more than half of inequalities

$$\varphi_1(x) > 0, \dots, \varphi_q(x) > 0.$$

Then the model of restriction $f_1(x) \leq 0$ will be the following inequality:

$$\sum_{i=1}^q \text{Sgn } \varphi_i(x) < 0,$$

where

$$\text{Sgn } t = \begin{cases} +1, & t > 0, \\ -1, & t \leq 0. \end{cases}$$

The application of committees to identification problem is based on that fact [1], that if $A \cap B = \emptyset$, $|A| < +\infty$, $|B| < +\infty$ and Φ is the class of all affine functions, then the system (4) has the committee.

3. THE FINDING OF OPTIMAL VECTOR

If the restrictions (1) of problem (2) are inconsistent, then we may understand this problem in sense of maximization of criterium $f(x)$ on the maximal consistent subsystems of (1). The other way is based on understanding of committee $K = \{x^1, \dots, x^q\}$ of system (1) like mixed strategy where x^i is pure strategy which used with probability $\frac{1}{q}$ ($i=1, \dots, q$). In this case it is naturally, to maximize the mean value of f on the set of all such mixed strategies.

If \mathcal{K} is the set of all committees of system (1), then we may interpret the problem (2) like

$$\sup \left\{ \frac{1}{|K|} \sum_{x \in K} f(x) : K \in \mathcal{K} \right\}.$$

4. THE PACK "KVAZAR"

Let us consider the problems of software and given some examples of the applied problems solved. A pack of applied programs termed "KVAZAR" (complex of calculating algorithms for solving pattern recognition problems) has been elaborated at the Institute of Mathematics and Mechanics of the Ural Scientific Centre of the Academy of Sci. of the USSR, to be used for solving a wide range of practical problems, basing on the technique of pattern recognition and multivariate statistics [2]. The sphere of application of the pack is quite extensive and includes automatic management systems, economics, sociology, biology, medicine, etc. Some of the examples of the problems which can be solved through the use of the above-mentioned pack are as follows: problems of industrial quality control, forecasts and prediction problems, classification, typology and others.

The pack features methodological orientation and is intended for use in solving the following pattern recognition problems:

- a/ problems of "training by tutor" or discriminant analysis;
- b/ problems in the taxonomy;
- c/ problems of selection of an informational subsystem of signs, which is sometimes formulated as the problem of presentation of multivariate data in a space of smaller dimensions (as, in particular in R^2); this with possible yield of the results on a graph-plotter or a graphical display screen.

Besides, the above pack will be found useful in solving the problems of multi-factor regressional analysis and a number of auxiliary problems, as well:

- 1/ Editing the basic data;
- 2/ transformation of vectors with components in the form of any real numbers into binary vectors, consisting of zeroes and units;

3/ transformation of the system of quantitative signs as per the main components-method;

4/ forming the training and checking sampling tasks for the problem of "training by tutor";

5/ mapping and yielding of histograms of distributions of quantitative values of signs onto the graph-plotter unit.

Realized in the pack to cater for the problem of "training by tutor" are the following four algorithms; 1/ algorithm to be used for construction of decision rules in the form of a homogeneous majority committee [3, 4, 5]; 2/ algorithm for construction of discriminating committees with seniority logics [6], 3/ one recurrent algorithm for discrimination between the convex shells of sets [7] and 4/ training algorithm founded on the concepts of the well known method of potentials [8]. Besides, provided in the pack is the possibility of constructing the committee discriminating functions in the conversational mode with the help of a graphical display unit. The advisability of such an approach to the solution of said problem is due to the difficulties encountered in the construction by automated means of affine discriminating committees in R^2 (even in the case $n=2$). In general, the concept of construction of committees through the use of a graphical display consists in the following:

Let us assume that sampled-out are the representatives of two classes in the form of a finite set of n -sized vectors. The mapping of this set is made in space R^2 (e.g. assisted by the main components method). The mapped set is then displayed in light on the screen. Here different symbols are used to map the objects of various classes, e.g. "+" or "0".

The task of the operator at the control panel with display unit consists in obtaining (in the dialogue mode) the image on the screen, where the classes may be discriminated in the more simpler way. Thenthrough the use of a "light plume", drawn are the committees' planes (which are straigh in the R^n space) according to the definition of the committee, while the mathema-

tical software of the Grafor system helps to obtain the coordinates of respective vectors - the committee members.

Realized in the pack to cater for the solution of problems in taxonomy there are three algorithms dealing with automatic classification: 1/ algorithm of the type "correlational pleiads" [9, 10]; algorithm known as "SPECTR" [11, 12], operating with potential functions and algorithm "TAKLIN", which uses the "statement of a problem" technique, concerned with the analysis of a system of linear inequalities and employing a requisite mathematical apparatus [13].

The first of the above algorithms, besides its principal purpose, is used in the pack also for the solution of automatic forming of training and checking tasks.

For the solution of the problem of selection of an informational subsystem of signs the above pack employs the well known method of "random search with adaptation" (RSA) [14] and a single heuristic procedure of selection of binary signs, based on the analysis of the frequency of encountered singular values of signs in the selected samples of different classes. When operating with a pair of classes, of interest is the study of histograms of distribution of the values of quantitative signs. The pack helps to form the histograms for each individual class, yielding them onto the graph-plotter, which gives an image of histogram of different classes in the same coordinate axes, which are distinguishable by their colour.

The above pack termed "KVAZAR" employs the formula translating system "FORTRAN" to be used with computer type BESM-6 and is intended to operate under the auspices of the monitory system "DUBNA" and operational system "DISPAK" or "DIAPAK". The "KVAZAR" system comprises the following component elements:

a/ set of functional modules used to realize the algorithms and procedures;

b/ monitoring program consisting in the, so called, pack system modules;

- c/ language of input directives;
- d/ accompanying documentation.

The "KVAZAR" pack, as to the possibility of replenishment with new modules, can be considered as an open system. Realized in the pack is the principle of programs segmentation, thus contributing to the solution of "memory" problem in case of a large module-containing system. Due to such organisational feature, the computer operating memory continually maintains in the course of problem solution only the residential module of the monitor, while the rest of the modules are introduced into the memory compartment, only when they endowed with control functions.

As mentioned above, the planning unit determines the inter-module connections, as to control functions. The linkage as per the data contained in the pack is ensured by using the common memory units "COMMON", magnetic drum and a temporary data bank, organized in the course of the problem solving through the use of a magnetic disc on the basis of employing a system of direct access to the external memory means [15].

The library of modules at the present time, besides the above-mentioned three-systems type, comprises also 33 functional modules (of which 18 are principal and 15 auxiliary). The modules are kept in store in pre-translated form (employing the load language). Besides, stored in the external memory there is a library of modules in the form of textural files, which may be edited with the help of the "EDITOR" program of the monitoring system.

The job generation for the pack is ensured through the use of the input directives language. In this case, with one address put into the pack in its proper operating mode a whole complex of problems can be solved, in accordance with the set of directives, included in the job.

The list of directives used in the pack (complete and in abbreviated form) is given below, so as to give a fuller scope of the functional possibilities of the latter:

- Vectors with real numbers;
- Vectors with binary components;
- Information input through punched-cards;
- Exception of rows in masking;
- Exception of real signs in masking;
- Exception of binary signs in masking;
- Single-factor polynomial regression analysis;
- Multi-factor linear regression analysis;
- Multi-factor regression analysis of the 2nd order;
- Transformation of the real signs system as per the method of main components;
- Mapping of a set of vectors onto the plane of main components;
- Formation and yield of histograms onto the graph-plotter (for a pair of classes);
- Taxonomy with the use of TAXE-module;
- Printing of taxson graph;
- Taxonomy with the use of TAKLIN-algorithm;
- Taxonomy with the use of SPECTR-algorithm;
- Informational subsystem of real signs (task or job for the search of same);
- Informational subsystem of binary signs;
- Formation of the training and checking samples via the taxonomy;
- Yield of information on the composition of the training and checking samples onto the punched-cards;
- User given tasks for training and checking samples;
- Samples previously formed by the pack;
- Training through the use of homogeneous committees technique;
- Training through the use of "potentials" method;
- Training through the use of recurrent algorithm of discrimination between convex shells;
- Recording of a decision rule on magnetic tape;
- Pattern recognition with the help of a previously obtained decision rule;

- Mapping of committees on a plane with the use of a graphical display;
- Pack performance monitoring.

The above-described "KVAZAR" pack has been duly tested and approved in the solution of a number of practical problems. In particular, the pack has helped, with good results, to solve the problem of diagnosis in the medical field: e.g. acute disturbances in the brain blood-circulation (cerebral apoplexy). Two kinds of ictus paraliticus are discerned: cerebral hemorrhage and the so called "malacia of the brain". Although the direct causes for these ailments are different, their symptoms are in many cases quite identical. As a rule, in both cases the illness is of a spontaneous nature and develops rapidly, sometimes causing disorders of conscience, paralyses and other characteristic symptoms. Due to this, an accurately made diagnosis with patients suffering from acute ictus paraliticus is in itself quite a complicated problem. Basing on the data taken from literature [16], about one half of the diagnostic decisions taken by physicians, both under the polyclinical or emergency first-aid conditions are erroneous. Even the physicians at the specialized neurological clinics make up to 20-25 % of erroneous diagnoses, this in the course of the first 24 hours of illness.

Hence the task of the machine-assisted diagnostics of the character of the cerebral apoplexy was taken in hands by the Institute of Mathematics and Mechanics of the Ural Scientific Centre of the USSR Academy of Sciences in collaboration with the Sverdlovsk Regional Informational and Computational Centre.

The job of training the automatic computer was preceded by the solution of the problem of taxonomy, as presented by the collaborators of the latter Centre, in the form of a data file containing the patients' illness charts with thoroughly verified diagnoses. Basing on the taxonomical results, the samples of training and checking techniques were mapped, containing 3 and 32 vectors, respectively. The training was ef-

fects through the use of an algorithm of construction of committee making decision rules. As the result of training, five decision rules were obtained, giving a high percentage of correct recognition (92 to 95 %) of the check vectors (and 100% - true recognition of the training vectors used). The above decision rules were further used for operational diagnostics.

The medical patient-treating services in the city of Sverdlovsk over a period of four years (1974 to 1980) have submitted to the Institute of Math. and Mech. of the Ural Scie. Centre more than 3,000 diagnostic requests, covering about 1,400 patients suffering from cerebral apoplexy. The accuracy of diagnostical response was within 88 to 92 %, which is somewhat higher than the result obtained by specialized physicians of neurological clinics.

At the present time the studies based on the use of pattern recognition techniques for diagnosing cerebral apoplexy are being continued. The principal aim of the above studies is to create a system capable of differential diagnostics of the above-mentioned ailment and yielding accurate answers in regard of the character and localization of the pathological process.

Besides the described medical problem, the "KVAZAR" pack has also been useful in solving the following problems: forecasts concerning the quality of industrial production, classification of automotive transport outfits in the Sverdlovsk district, forecasts as to the level of the ambient air pollution in the deepsunk quarries, classification of some biological populations, forecasts pertaining to the outcome of delivery (in child-birth situations) in regard of the newly-born infants, etc.

R E F E R E N C E S

- [1] Mazurov V.I.D.
Inequality committees and recognition problems
Cybernetics (Kiev), 1971, N^o 3, p.140-146.
- [2] Mazurov V.I.D., Kazantsev V.S., Beletsky N.G., Mezentsev S.V., Sachkov N.O.
The "KVAZAR" pack of applied programs in pattern recognition (version 2): Inform. Materials on mathematical software.
Sverdlovsk: Inst. Math. and Mech., Ural. Sci. Center of Acad. Sci. USSR, 1979, N^o 33, - 74 p.
- [3] Mazurov V.I.D.
On the construction of the committee of convex inequalities.
Cybernetics (Kiev), 1967, N^o 2, p. 56-59.
- [4] Mazurov V.I.D.
Method of committees used for pattern recognition
Sverdlovsk: Inst. Math. and Mech., Ural. Sci. Center of Acad. Sci. USSR, 1974, N^o 6.-165 p.
- [5] Kazantsev V.S., Tjagunov L.I.
Algorithm for committee - based recognition: Optimization programs.
Trans. of the Inst. of Math. and Mech., Ural. Sci. Center of Acad. Sci. USSR (Sverdlovsk), 1974, N^o 5, p.61-78.
- [6] Osborne M.L.
The seniority logic - a logic for a committee machine.
IEEE Trans. Comput., 1977, vol. C-26, p.1302-1306.
- [7] Kozinetz V.N.
Recurrent algorithm for discriminating between two sets
In: Algorithms for pattern recognition training/
Ed. by V.N. Vapnik. Moscow: Sovjet Radio, 1972, p.43-50.

- [8] Arkadiev A.G., Braverman E.V.
Training the machine for classification of objects
Moscow: Nauka, 1971.- 191 p.
- [9] Terentiev P.V.
Method of correlational Pleiads.
Vestnik of Leningrad State Univ. Biological Series,
1959, N^o 9, p. 137-141.
- [10] Tyagunov L.I., Sheveleva L.I.
Taxonomy Algorithm in the Hemming's Metrics
In: Optimization programs: (Optimum Planning). Sverd-
lovsk: Inst. of Math. and Mech., Ural. Sci. Center of
Acad. Sci. USSR, 1976. N^o 5, p. 79-87.
- [11] Dorofeiuk A.A., Lumelsky V.Ya.
Realization of Algorithms for training without a tutor
the pattern recognition techniques using a computer.
In: Pattern recognition training algorithms/ Ed. by V.N.
Vapnik. Moscow: Sovjet Radio, 1972 p. 181-198.
- [12] Sachkov N.O.
On the software for taxonomy problem solving.
In: Optimization programs: Planning of ore-mining and
metallurgical production. Sverdlovsk: Inst. of Math. and
Mech., Ural. Sci. Center of Acad. Sci. USSR, 1977, N^o 7,
p. 151-158.
- [13] Kazantsev V.S.
Taxonomy algorithm employing linear inequalities.
In: Methods for nonstationary problems in mathematical
programming. Sverdlovsk: Inst. of Math. and Mech., Ural.
Sci. Center of Acad. Sci. USSR, 1979, N^o 29, p. 120-125.
- [14] Lbov G.S.
Selection of an effective system of dependent signs.
In: Computational Systems. Novosibirsk, 1965, N^o 19,
p. 21-34.

- [15] Mitiusheva L.L., Ponomareva L.S.
"DUBNA" monitoring system catering for "DISPAK" software:
(Operating Instructions).
Sverdlovsk: Ural. Sci. Center. of Acad. Sci. USSR, 1976
196 p.
- [16] Gurvitz T.V., Iovlev B.V., Tonkonogii I.M.
Tabular computational methods in diagnostics or cerebral
appoplexy cases and forecasts on the issues.
Leningrad: Medicine, 1976. - 200 p.

ON PROGRAM PACKAGE DISPRO FOR SOLUTION OF
DISCRETE OPTIMIZATION PROBLEMS

V.S. Mikhalevich, I.V. Sergienko, T.T. Lebedeva,
V.A. Roshchin, A.S. Stukalo, V.A. Trubin, N.Z. Shor

(Kiev, USSR)

An application program package (APP) DISPRO [1] developed at the Institute of Cybernetics of the Ukrainian Academy of Sciences is intended for solution of a wide range of general and special problems of discrete optimization. It can operate in the batch-processing and interactive modes in compatible models of ES computers (ES-1022 and upwards) under the operating system OS ES (versions 4.1 and 6.1) in MVT and MFT modes. Various parts of the package are implemented in Assembler, FORTRAN-IV, PL/1. An overall requirement for this software is 1800 KB.

When studying many theoretical and practical problems the optimization problems arise in which all or a part of variables must assume integral values or values from a given discrete set. Problems of such type are solved in drawing up plans of operation, reconstruction, development or location of plants and industries of the national economy, in designing equipment and machines, in transport control, in resource-constrained job scheduling and in many other cases.

At present a sufficiently developed theory of discrete optimization is devised which covers studies on the structure and properties of different classes of problems, methods of their solution, methods of estimating the labor intensivity of solution and other aspects. Methods of solution of both general and special problems of discrete (for the most part, linear) programming are developed.

Despite the fact that the theory of the existing methods of discrete optimization is sufficiently developed they are not

suitable for solving a wide range of practical problems. Besides, results of the complexity theory [2] for optimization problems show that the hope for creation of exact methods ensuring the solution of discrete optimization problems in an admissible time is an illusion. Though there are examples of solution of such problems with several hundreds of variables, in the general case there is no confidence in obtaining the optimal solution of problems with 30-40 integer variables. Most of special discrete optimization problems turn out to be theoretically as complex as the problem of linear integer programming in the general formulation. The analysis carried out on 9000 special problems of the control theory [3] showed that approximately 9% of these problems are effectively solvable, 77% fall into the class of general-purpose (NP-complete) problems and 14% are not classified by complexity so far. Naturally, the simplest of them fall within the class of efficiently solvable problems. Probably this is the case for all other problems of the discrete optimization.

From these facts it is inferred that it is necessary to develop and use such algorithms which make it possible to obtain approximate solutions in the cases when exact solutions are not possible to construct. It is known that the algorithms with different efficiency solve individual classes of problems and in this case it is difficult to predict their a priori efficiency for special problems. Therefore it is advisable to have the possibility of testing several algorithms which implement different types of methods. The larger set of algorithms the more probability of obtaining feasible solutions with minimum expenditure.

From this it follows that application program packages oriented towards solution of a broad class of discrete optimization problems should comprise a large set of algorithms which make possible the construction of exact and approximate solutions of general and special problems.

At present there are a number of home and foreign application program packages for solution of discrete optimization problems in ES computers in which only one of known methods - the branch-and-bound method - is implemented. The APP LP ASU (linear programming of management information system) is the most popular example of such package.

Information on APPs for solution of integer programming problems designed in the recent decade abroad (mostly in the USA) as of the middle of 1977 is given in [4]. All packages are based on the branch-and-bound method, use efficient programs of linear parametric programming, make possible the construction of approximate solution (in a heuristic sense) and give a user some possibilities of solution process control. These programs do not assure the obtaining of the optimal solution in an admissible time in problems with the number of integer variables exceeding 20÷40.

By its possibilities for solution of a large class of discrete optimization problems the APP DISPRO has no direct analog in the world practice of designing such software. Modern efficient algorithms for solution of general and special discrete programming problems are implemented in this package.

Compared to the existing program packages for solution in ES computers of mathematical programming problems the package DISPRO has the following advantages:

- The package can operate in the interactive mode;
- A reliable syntactical and logical data check is performed and dialog means of data preparation and maintenance are developed;
- The package may be enlarged and expanded to include application modules in Assembler, FORTRAN-IV, PL/1;
- Preparation of data in a natural format convenient for a user and efficient means for creation, correction and storage of the sets of data (initial, intermediate, resulting) are realized;
- The automatic selection of a method of solving the dis-

crete optimization problem and the choice of the method among a set of methods realized in the method package by user's instructions are provided.

The use in APP DISPRO of the dialog means for solution of discrete optimization problems significantly increases the efficiency of a number of employed methods especially in cases of large-scale problems and when it is required to study the dependence of the obtained solutions on method parameters and (or) initial data. Dialog means may be used for correction of mathematical models of the problems being solved as well.

Consider mathematical models and methods incorporated in the APP DISPRO.

To solve completely integer linear programming problems of the general form

$$\min\{cx \mid Ax \leq b, x \geq 0, x \text{ is an integer vector}\} \quad (1)$$

the package contains modules realizing the branch-and-bound method [5], returning algorithm of the cutting-off method [6] (for exact solution of these problems), the third Gomory algorithm with the use of dynamic programming [7] which is intended for exact or approximate solution of integer programming problems with integer condition matrices. The package also contains the methods of local optimization - the fall-off vector method and the method of direction neighborhoods [8,9] for approximate solution of problems of the form (1). For the approximate solution of large-scale integer knapsack problems a random search algorithm is realized.

The use of many approximate methods among which are methods of the fall-off vector and the direction neighborhoods presuppose the preliminary assignment of some initial approximations which are feasible solutions. Initial approximations may be chosen either by a user on the basis of the distinguishing features of the problems being solved or with the help of a number of special algorithms. For this purpose the algorithms

of search for feasible solutions to problems of the integer linear programming (ILP) of the form (1) based on the methods of controlled random search and direction neighborhoods as well as on the geometrical approach are implemented in the package [9].

The following parametric model of the integer linear programming is represented in the program package DISPRO

$$\min\{cx \mid Ax \leq b + tb', \ x \geq 0, \ x \text{ is an integer vector}\} \quad (2)$$

where the parameter $t \in [\xi, \varphi]$, and ξ, φ are specified real numbers.

The algorithm [10] is realized which uses the ideas of the method of direction neighborhoods and is intended for the approximate solution of problems of the form (2).

For the approximate solution of quadratic problems of the integer programming

$$\min\{dx + x'Cx \mid Ax \leq b, \ x \geq 0, \ x \text{ is an integer vector}\}$$

the method of fall-off vector is realized in the package.

The program package DISPRO contains the algorithm [11] which realizes the necessary stage of solution of problems of parametric integer programming of the form: to find $\text{extr } f(x)$ under restrictions

$$(A + tA_1)x = b + tb_1, \quad (3)$$

where A, A_1 are integer $(m \times n)$ -matrices; b, b_1 are integer m -vectors; x is the desired integer n -vector; t is the integer parameter $(-\infty \leq t \leq +\infty)$. This algorithm consists in determination of the values of parameter t for which the system (3) has no integer solutions for certainty. It permits of finding the values of parameter t for which the solution of system (3) is unique (if it exists); this eliminates the need to prove the optimality of obtained feasible solutions for these values of

parameter and considerably simplifies the process of solution of the stated optimization problems.

To solve ILP problems with Boolean variables of the form

$$\min\{cx \mid Ax \leq b, x = (x_1, \dots, x_n), x_j = 0 \text{ or } 1, j = \overline{1, n}\} \quad (4)$$

the package DISPRO contains 8 modules realizing different methods. Among these are the Balas method [12] and the method [13] based on the use of concepts of the method of sequential analysis of variants [14,15] which are used for exact solution of problems of the form (4) and also the method of fall-off vector [9] for their approximate solution. To search for initial approximations necessary for using the method of fall-off vector and a number of other approximate methods the algorithm for controlled random search for feasible solutions of problems of the form (4) is included.

The program package DISPRO contains also a model of a large-scale knapsack problem with Boolean variables, and the following three algorithms for its solution are realized: an algorithm of the lexicographical search for exact (or approximate) solution, an algorithm of random search for an approximate solution and the modified Balas algorithm for the approximate solution of the given problem.

The program package contains 9 modules which realize a number of methods used for solution of practically integer linear programming problems of the form

$$\min\{cx + dy \mid Ax + By \leq b, x \geq 0, y \geq 0, x \text{ is an integer vector}\} \quad (5)$$

These are the branch-and-bound method [5], the Benders method [16] and the returning algorithm of the cutting-off method [6] for exact solution of such problems, and the methods of fall-off vector, direction neighborhoods and one algorithm [9] using the concepts of the Benders method which are employed for the search for an approximate solution of problems of the type (5). To find initial approximations necessary for realization of the

algorithms of fall-off vector and direction neighbourhoods the package contains the algorithm relied on a geometrical approach.

The problems of type (5) in which the additional condition $x \leq 1$ is imposed on vector x the algorithm of random search for approximate solutions to these problems is realized.

The package DISPRO also contains 10 modules for solution of problems of special classes. These modules permit of the construction of exact or approximate solution of some problems of production allocation, scheduling theory, resource allocation, partitioning, packaging and covering of sets. Most of them realize a scheme of branches and bounds with the use of structural algorithms, for computation of functional bounds. Modules are organized in such a manner that they make possible the rapid construction of a feasible approximate solution using the remaining computation time either for improvement or for proving of its optimality. Below are given models of problems of the considered classes and brief characteristics of the methods.

1. THE PROBLEM OF ONE-STAGE LOCATION OF PRODUCTION

We are required to find

$$\min \sum_{i=1}^m \sum_{j=1}^n d_{ij} x_{ij} + \sum_{i=1}^m f_i(X_i) \quad (6)$$

subject to

$$\sum_{i=1}^m x_{ij} = B_j, \quad \sum_{j=1}^n x_{ij} = X_i, \quad x_{ij} \geq 0, \quad i = \overline{1, m}, \quad j = \overline{1, n} \quad (7)$$

where

$$f_i(X_i) = \begin{cases} a_i X_i + b_i (a_i, b_i \geq 0), & \text{if } X_i > 0, \\ 0, & \text{if } X_i = 0, \quad i = \overline{1, m} \end{cases} \quad (8)$$

Here $f_i(X_i)$ is the function of production costs at the i -th point of possible location; B_j is the j -th customer's need for the product; $D=(d_{ij})$, $X=(x_{ij})$ are matrices of specific transportation costs and unknown amount of transportation; X_i is an unknown productive capacity at the point i ; $i=\overline{1,m}$, $j=\overline{1,n}$.

Problems of location of production of a homogeneous product with concave piecewise linear functions of production costs and the problem of unification of a number of products are formulated in the form (6)-(8). With regard for specific features, it may be transformed into the problem of partially integer linear programming [17]:

$$\sum_{i=1}^m \sum_{j=1}^n C_{ij} y_{ij} + \sum_{i=1}^m b_i y_i \rightarrow \min, \quad \sum_{i=1}^m y_{ij} = 1,$$

$$0 \leq y_{ij} \leq y_i, \quad y_i = 0 \vee 1, \quad i=\overline{1,m}, \quad j=\overline{1,n}$$

Here

$$C_{ij} = (d_{ij} + a_i) B_j, \quad y_{ij} = B_j x_{ij}, \quad i=\overline{1,m}, \quad j=\overline{1,n}.$$

To solve this problem the branch-and-bound method with a one-sided scheme of branching with respect to variables y_i and with construction of the lower bound of the functional by the approximate solution of the corresponding dual problem of the linear programming, is realized. The bound is computed in the time $O(m^2 n)$, the required memory is $O(mn)$. The approximate method relies on the principle of dynamic realization the essence of which is the possibility of introducing the partial ordering of variables of the problem being solved and its use for the construction of efficient exact and approximate methods of solution. This principle is valid for a number of generalizations of the allocation problem, in particular, for problems of the synthesis of networks with convex cost functions. Besides the lower bound it permits of constructing a feasible solution to the integer problem the functional value of which is the upper bound for the optimal solution.

2. THE PROBLEM OF TWO-STAGE LOCATION OF PRODUCTION

It is required to find

$$\min \sum_{i=1}^m \sum_{j=1}^n a_{ij} x_{ij} + \sum_{j=1}^n \sum_{k=1}^p b_{jk} y_{jk} + \sum_{i=1}^m f_i(X_i) + \sum_{j=1}^n g_j(Y_j) \quad (9)$$

subject to

$$\sum_{j=1}^n y_{jk} = B_k, \sum_{i=1}^m x_{ij} = \sum_{k=1}^p y_{jk} = Y_j, \sum_{j=1}^n x_{ij} = X_i \quad (10)$$

$$x_{ij} \geq 0, y_{jk} \geq 0, i = \overline{1, m}, j = \overline{1, n}, k = \overline{1, p}.$$

Here $f_i(X_i)$, $i = \overline{1, m}$, $g_j(Y_j)$, $j = \overline{1, n}$, are cost functions of the form (8) for location of production of the first and second levels at points i, j , respectively; B_k is the k -th customer's need for the homogeneous product; X_i, Y_j are unknown capacities of productions of the first and second levels; $a_{ij}, b_{jk}, x_{ij}, y_{jk}$ are specific transportation costs and unknown amounts of transportation respectively between points of location of the first and second levels and between the second level and points of consumption; $i = \overline{1, m}, j = \overline{1, n}, k = \overline{1, p}$.

To this model is reduced the problem of location of production of homogeneous products with convex piecewise linear functions of production costs in case when finished products are obtained as a result of the processing (or mining) of raw materials or semifinished items at two stages. For example, ore mining - extraction of metal - consumption. This problem and the one-stage problem have many overlappings. For its solution an algorithm is realized which shares the same principle as the foregoing one; the bound is computed in time $O((m+n)^2 p)$, the required memory is $O(n(m+p))$.

3. THE PROBLEM OF OPTIMAL ALLOCATION OF LIMITED RESOURCES

We are required to find

$$\max \sum_{j=1}^n f_j(x_j)$$

subject to

$$\sum_{j=1}^n g_{ij}(x_j) \leq g_i, \quad x_j \in X_j, \quad i=\overline{1, m}, \quad j=\overline{1, n},$$

where X_j is an assigned discrete set of admissible values of x_j ; $g_{ij}(x_j)$ are nonnegative functions which characterize costs of the i -th resource when choosing the value of the j -th variable; g_i is the amount of the i -th resource; $i=\overline{1, m}$, $j=\overline{1, n}$. Examples of problems of this type are: problems of optimal redundancy, of forming of orders booked, of dynamic allocation of investments, etc. For their solution the method of sequential analysis of variants [18] is realized which permits, depending upon allotted memory, of construction of exact and approximate solutions.

4. NETWORK RESOURCE ALLOCATION PROBLEM WITH INDEPENDENT SEQUENCE OF OPERATIONS

A set of m jobs must be performed. The i -th job, $i=\overline{1, m}$, consists of n_i operations numbered 1 to n_i . The time to perform each operation is an integer represented by d_{ij} for the j -th operation of the i -th job. A set of l different resources is available; r_{ij}^k is the amount of the k -th resource required by operation ij during its execution; $R_{kt} \geq 0$ is the amount of the k -th resource available in time period t (t is an integer, $1 \leq t \leq T$). We consider the problem that meets the following conditions: 1. Each operation requires only a single type of the resource. 2. $R_{kt} = R_k$, $t=\overline{1, T}$, - levels of the available resources are constant in time. 3. No interruption of the operation

execution is allowed. Once the operation is started, it must be executed until completed in d_{ij} time units. 4. Let t_{ij} represent the start time for operation ij . For each operation ij early \underline{t}_{ij} and late \bar{t}_{ij} dates of the start of completion, are specified, i.e. for the condition $\underline{t}_{ij} \leq t_{ij} \leq \bar{t}_{ij}$ must be satisfied. 5. Technological constraints of succession of operations are expressed by conditions of the form $t_{ij} \geq t_{ik} + d_{ik}$ if in the sequence of operations of the job i the ij - is directly followed by the operation ik . 6. Let $I_t = \{ij | \underline{t}_{ij} \leq t \leq \bar{t}_{ij} + d_{ij}\}$ be a set of operations performed in time period t ; $f_{ij} = t_{ij} + d_{ij}$ be the completion time of operation ij ; $f_i = t_{ij} + d_{ij}$ the completion time of job i . At each time instant t the resource constraints

$$\sum_{i,j \in I_t} r_{ij}^k \leq R_{kt}, \quad k=\overline{1, \bar{l}}, \quad 1 \leq t \leq T \text{ must be met.}$$

7. A set of values for $\{t_{ij}\}$ that satisfies the given condition defines a job schedule. An objective function $z = \sum_i g_i(f_i)$ where

$g_i(f_i)$ are nondecreasing functions is specified on a set of schedules. We are required to find a schedule with the minimal value of z .

Let us write a linear integer model of the problem. Let

$$x_{ij}^k = \begin{cases} 1 & \text{if the operation } ij \text{ starts in time period } t, \\ 0 & \text{otherwise;} \end{cases}$$

$C_i^t = g_i(t + d_{in_i})$, $T < \infty$ be a total duration of the planning interval. We are required to find

$$\min \sum_{i=1}^m \sum_{t=1}^T C_i^t x_{in_i}$$

subject to

$$\sum_t x_{ij}^t = 1; \sum_{ij} \sum_{\tau=\max(0, t-d_{ij}+1)}^t r_{ij}^k x_{ij}^{\tau} \leq R_{kt}, \quad k=\overline{1, L}, \quad t=\overline{1, T};$$

$$\sum_t x_{ij}^t + d_{ij} \leq \sum_t x_{i(j+1)}^t, \quad i=\overline{1, m}, \quad j=\overline{1, n_i-1}; \quad x_{ij}^t = 0 \vee 1 \quad \forall i, j, t.$$

Such problems arise in highway engineering scheduling, in equipment mounting, in the computer-assisted solution of problems in the batch mode of operation. To solve the described problem the branch-and-bound method is realized in the package with the use of the method of Lagrange multipliers in discrete programming [19]. The bounds are constructed by dualizing the resource constraints. The bound problem is solved by nonsmooth optimization methods for finding the maximum on Lagrange multipliers and by dynamic programming methods for solution of the problem with fixed multiplier values in each iteration of the search. The module is oriented towards approximate solution of the original problem with the use of variable branching strategy and is intended to solve the problems with $L.T < 900$, $n < 300$ where n is the total number of operations in the network.

5. THE NETWORK JOB SCHEDULING PROBLEM

We are required to determine the minimal time and the order of performance of the given set of jobs with precedence constraints in the form of an acyclic oriented graph, and with identical (unit) realization durations. Each operation is associated with the fixed amounts of different resources required by it. Given the step time functions of the integer arguments of amounts of each type, only those schedules are admissible which preserve precedence relations and employ resources of each type in the amount not exceeding the available one. For solution of this problem the dynamic programming method is chosen. The corresponding module is realized in such a manner that an approximate solution is constructed in case of lack of the memory (internal and external) [20].

6. THE SCHEDULING PROBLEM WITH ACYCLIC OPERATIONS NETWORK

This problem is generalization of the above problem but differs from it in the following: 1/ Durations of indivisible operations are integers but not necessarily the same values; 2/ A criterion by which the schedule is optimized consists either in minimizing maximum resource-consumption time (problem 6 (A)) or in minimizing job completion time at the resource level specified in time (problem 6 (B)); 3/ The amount of types of resources is equal to 1. We are required to find a schedule, admissible by the given criterion, under which each job is performed without interruptions and a set of jobs meets the precedence requirements specified in the form of an acyclic graph whose nodes correspond to jobs. If in the graph there is an arc (i, j) job i must be completed before job j is started.

Such problems arise in the automated network scheduling and control systems, in the organization of computational process in parallel computer systems, etc. For exact or approximate solution of problem 6(A) the branch-and-bound algorithm with computation of the lower bound by the method described in [21] is developed. The problem 6(B) may be solved by solving a sequence of problems with various restrictions on the completion time of all jobs.

7. THE PROBLEM OF SPACE-TIME DISTRIBUTION OF MACHINES AMONG JOBS

It falls in a special class of problems of allocation of limited resources and consists in the following. A set of partially ordered jobs fixed in space and time and a set of machines with the help of which the jobs must be performed (machines are not always interchangeable) are specified. The performance and its cost depend on the type of the machine and the job. A change to some job far removed in space is connected with costs and duration of dismounting-mounting and displacement of machines. We are required to find an optimal schedule of using the machi-

nes for all jobs.

Let C_{ij} be costs of job j performed by machine i ; a_j be a coefficient of importance of the j -th job; b_i - the total operation cost of machine i ; $f(s_i)$ - costs of moving machine i along the route s_i which connects some sequence of jobs; $x_{ij} = 0 \vee 1$ - Boolean variables assuming the value 1 if in the problem solution the machine i performs the job j , and 0 otherwise; $i = \overline{1, m}$, $j = \overline{1, n}$. The problem consists in searching for the lexicographic minimum:

$$\text{lex max}(z_1, -z_2, -z_3). \text{ (or } \text{lex max}(z_1, -z_3, -z_2)\text{)}$$

subject to

$$\sum_{i=1}^m x_{ij} \leq 1, \quad j = \overline{1, n}; \quad x_{ij} + x_{ik} \leq 1;$$

$$\sum_{i=1}^m x_{ij} \geq \sum_{i=1}^m x_{ik}, \quad \text{if } j \prec k; \quad x_{ij} = 0 \vee 1, \quad i = \overline{1, m}, \quad j = \overline{1, n}, \quad (j, k) \in U,$$

where U is a set of job pairs which cannot be performed with one machine, \prec is a partial-order relation for the jobs;

$z_1 = \sum_{j=1}^n a_j \sum_{i=1}^m x_{ij}$ is the weighted amount of jobs being performed;

$z_2 = \sum_{i=1}^m \sum_{j=1}^n C_{ij} x_{ij} + \sum_{i=1}^m f(s_i)$ are the total costs of servicing

the jobs; $z_3 = \sum_{i=1}^m b_i \sum_{j=1}^n x_{ij}$ are the total costs of using the

set of machines under operation. To solve this problem the branch-and-bound method and an approximate method with polynomial number of computations [22] are realized in the package.

8. THE DISTRIBUTION PROBLEM WITH BOOLEAN VARIABLES

It is required to find

$$\max \sum_{i=1}^m \sum_{j=1}^n a_i b_j x_{ij}$$

subject to

$$\sum_{j=1}^n x_{ij} = 1, \quad \sum_{i=1}^m c_{ij} x_{ij} \leq d_j; \quad x_{ij} = 0 \vee 1; \quad i = \overline{1, m}, \quad j = \overline{1, n}.$$

A number of practical problems, in particular the problem of allocation of program modules in multilevel computer memory, are reduced to such model. For its approximate solution the method of fall-off vector taking account of distinguishing features of the problem is realized.

9. THE SET PARTITIONING PROBLEM

We are required to find

$$\max \sum_{j=1}^n c_j x_j$$

subject to

$$\sum_{j=1}^n a_{ij} x_j = 1, \quad x_j = 0 \vee 1, \quad i = \overline{1, m}, \quad j = \overline{1, n},$$

where $A = \{a_{ij} = 0 \vee 1\}$ is a specified matrix. This problem is often used in scheduling, transportation planning, circuit design, etc. For its solution the method combining the concepts of simplex-method and the branch-and-bound method is realized. It constructs a sequence of integer solutions along which the value of the objective function increases. To find a successive vertex a branch-and-bound scheme is employed in which the values of dual variables in the current vertex are used for computation of bounds [23, 24].

10. THE SET COVERING PROBLEM

We are required to find

$$\min \sum_{j=1}^n x_j$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \geq 1, \quad x_{ij} = 0 \vee 1, \quad i = \overline{1, m}, \quad j = \overline{1, n},$$

where $A = \{a_{ij} = 0 \vee 1\}$ is the specified matrix. Such problems arise in the same context as the partitioning problems. For its solution the method of implicit enumeration taking into account distinguishing features of the problem and four approximate methods are employed [25].

The package DISPRO contains a various set of optimization methods having the different efficiency on different classes of discrete programming problems. This is one of the advantages of the given package in comparison with existing ones. The speedy and simple introduction of new application modules into the package DISPRO gives the possibility of studying them experimentally.

Until recently, a comparison analysis of the efficiency of different modules presented difficulties because of the following circumstances: the absence of the comprehensive set of modules for engineer who develops new algorithms and modules; different ways of preparation and presentation of original data; orientation towards different computer classes. The package DISPRO permits of avoiding these difficulties.

In the process of development of the package DISPRO the experimental study of the efficiency of its methods for solving discrete optimization problems [1] which are of interest for package users was conducted. In particular, they give an idea of dimensions of the problems which may be solved by this package and of the mean dependence of the computation time on the problem dimension. These studies are also important because theoretical estimates of the problem solution laboriousness are unknown in most cases for discrete optimization methods. The analysis of experiments shows that many methods introduced into the package DISPRO operate efficiently on problems of the medium and large dimensionality. Some results of investigation

of the efficiency of a number of methods incorporated in the package are given in [1]. Results of the conducted experimental studies were the basis for the development of a unit of automatic choice of the method for solution of discrete optimization problem in the package DISPRO.

Basic principles of the development of the program package DISPRO are: package orientation towards a broad circle of users and efficient service support; close interaction between the package and operational system and programming systems; a high degree of automation of a computational process including organization, check and analysis of program execution, efficient means of preparation and input of the initial data and of output and display of the results; realization of modern modes of the package operation, including the interactive and batch processing modes; efficient solution of optimization problems (including the automation of the choice of solution method); the availability of documentation for the package sufficient for its production and operation. In accordance with the construction principles the main language, program and information facilities are: an input nonprocedure language, a control program, a set of application modules, a subject domain model, data bases, a service. The interaction between users and the package is conducted in the input language intended for description of initial data of optimization problems being solved, their parameters and results and for the solution process control in interactive and batch modes of operation. This language consists of operators with the help of which the package task is specified. By their functions the operators may be divided into three groups: the operators of job management, the operators of description of problems and data, the data manipulation operators.

The control program of the package referred to as monitor performs the following functions: interaction with OS ES (with a job scheduler, supervisor, access types); control of package operation in the batch and interactive modes and of package components operation; interaction with a user at different

stages of operation (input of the task for the package, output of messages and requests for the user when translating and performing jobs, reception of answers, solution of optimization problems in the interactive mode); input of tasks for the package from various input devices (from punch cards, the VDU display ES-7066, the typewriters); translation of input language operation; performance of task steps and the whole task; input, check, conversion, storage, correction and output of source data; storage and output of results to various output devices (an alphanumerical printer, display); organization of the automatic solution, execution and analysis of completion of application modules of optimization problem solution according to the specified model of subject domain; control of machine resources (immediate-access memory, input-output devices, direct access memory).

The DISPRO software has a hierarchic modular structure. The package kernel, the monitor resident, is at the upper stage of the module hierarchy, control modules of the job translator and executor are at the second stage, processing and executive modules of the job translator and executor are at the third stage. The lower stage contains service modules, application modules of optimization problem solution and modules of data manipulation (conversion, check and correction).

The monitor resident performs the following principal functions: package-OS ES interaction, initiation of control modules of the translator and executor, package operation mode control, analysis of program completion and processing of program interruptions, servicing (forming, loading, assigning) of the domain of monitor modules linking and organization of their interface, allocation of immediate-access memory in the package section, output of messages to the alphanumerical printer and display, operation of some control tables, exchange with internal devices (direct access memory, alphanumerical printer, punch cards, display).

Main functions of the input language translator are: input of package task operators from specified input devices, initiation of modules of translation of the input language operators, syntactical check of the operators and working out of corresponding diagnostics, initiation of the module of diagnostic message output, conversion of operators into the inner representation and generation of control tables of the executor, creation of the temporary library and its filling with sets of data, input and conversion of initial data into the internal format.

The executor (interpreter) of jobs is intended to organize the performance of jobs in the interactive (step-by-step performance) and batch processing modes. It realizes the following basic functions: Initiation of executive modules of the interpreter, control of the process of job performance, analysis of module functioning, completion and output of diagnostic messages, preparation and check of data for the job or job step, control of preparation or execution of application modules of optimization problem solution, organization of the output of source data and results of the solution.

The data management subsystem contains modules of check, correction and printout of results of alphanumerical printers in the specified output forms. The model of the subject domain, the discrete optimization problems and methods of their solution, includes the tables: of problems with computational schemes of optimization problem solution, of application module with names and parameters of the modules, of forms of source data and output of results of the problem solution.

The package DISPRO operates in one of the sections of the internal memory under control of OS ES. The package operation is initiated with the OS ES task where the operator *DD* is followed by the job to be performed by the package together with source data when inputting the job from punch cards or by source data only when inputting the job from the display. The name of the monitor resident is specified in the operator *EXEC*. At the

beginning of the package operation, the monitor resident is loaded into the internal memory and accepts the control. The resident is in the internal memory during all operations. At first it determines the mode of operation and the device for inputting the job to be performed by the package, loads the control monitor tables into the internal memory and opens the required data sets, then it initiates the control translator program and transfers the control to it.

The control program organizes the operation of modules, loads them into the internal memory, transfers the control to them and analyzes the completion.

At the batch mode of operation the job operators are input and translated until the job terminates. If the job consists of several steps then each step is presented by the translator as a unity of package operation. If errors are detected in the operators the control program of the translator sends diagnostic messages and the corresponding job step is marked as inexecutable. The control is transferred to the executor after translation of the whole job. In the dialog mode a step-by-step translation of job operators is accomplished: the control is transferred to the executor after translation of each step, and when the error is detected the diagnostic messages are issued and the package starts to operate in the mode of expectation of user's response. Upon completion of the translation operation the control is returned to the monitor resident which initiates the control program of the executor.

The executor control program organizes a step-by-step execution of the job to be performed by the package. It puts the operators on the queue for each executable step of the job and initiates the corresponding modules. In the batch mode the user's operation terminates after completion of all job steps. Inexecutable steps or abnormally ended steps are bypassed. In the interactive mode the executor's operation terminates after completion of the job step, the control is returned to the monitor resident which initiates the translation and execution of the next step by user's requirement. In the

batch-processing mode the user does not interfere in the process of problem solution. He receives information about job performance and problem solution with the help of diagnostic and information messages which are output to the printer or the display.

In the interactive mode the user may interfere in the package operation during the input, translation and execution of jobs and in the process of optimization problem solution; for this purpose the package has dialog application programs.

The application program package can be modified and extended. A modular structure of the package makes it possible to enlarge its input language, to include system and application modules, to realize in the next version the interface between the package and data base management systems and other APPs, to synthesize the package with programming systems in PL-1 and FORTRAN-IV languages.

The next version of the package DISPRO is designed on the basis of the considered principles and requirements on modern method-oriented APPs such as storage and analysis of data about package operation (results of the processing of these data may be used for the improvement of the package structure and software), large dimensionality of problems being solved, the operation reliability, industrial production of packages and their applicability for integrated data processing systems.

R E F E R E N C E S

- [1] Mikhalevich V.S., Sergienko I.V., Lebedeva T.T.,
Roshchin V.A., Stukalo A.S., Trubin V.A., Shor N.Z.
Application Program Package DISPRO Intended for Solution
of Discrete Programming Problems. - Kibernetika, 1981.
No.3, p. 117-137.
- [2] Karp R.M.
Reducibility of Combinatorial Problems. - Kiberneticheskiy
sbornik, 1975, vyp. 12, p. 16-36.
- [3] Rinnoy Kan A.H.G.
Scheduling. In: Annals of discrete mathematics. North-
Holland Publishing Company, 1979, No.5, pp.423-426.
- [4] Land. A., Powell S.
Computer codes for problems of integer programming. - In:
Annals of discrete mathematics. North-Holland Publishing
Company, 1979, No.5, pp.221-269.
- [5] Land. A.H., Doig A.G.
An automatic method of solving discrete programming prob-
lems. - Ekonometrika, 1960, 28, No.3, pp.497-520.
- [6] Chervak Ju.Ju.
The returning algorithm of the cutting-off method and of
the branch-and-bound method. - Ekonomika i mat. metody,
1978, 14, No.5, pp.1002-1005.
- [7] Shejnman O.K.
Duality and subadditive functions in integer programming.
- Ekonomika i mat. metody, 1980, 16, No.4. pp. 808-810.
- [8] Sergienko I.B.
Problems of the development of one approach to solving
discrete optimization methods in data processing systems
management information systems. - Upravl. sistemy i mashiny,
1974, No.6, pp.107-114.

- [9] Sergienko I.V., Lebedeva T.T., Roshchin V.A.
Approximation methods of solving discrete optimization problems. Kiev: Naukova dumka, 1980. 273 p.
- [10] Sergienko I.V., Lebedeva T.T., Filonenko N.V.
Approximate solution of problems of integer linear programming with a parameter in constraints. - In: Vychislitel'nye aspekty v paketakh prikladnykh program. Kiev: IK AN USSR, 1980. pp. 3-16.
- [11] Kozerackaja L.N.
On systems of linear diophantine equations with parameter Dokl. AN USSR. Ser. A, 1980, No.9. pp. 11-14.
- [12] Balas E.
The additive algorithm for solving linear programming problems with variables assuming values 0 or 1. - Kiberneticheskiy sbornik, 1979, vyp.6, pp.217-252.
- [13] Volkovich V.A., Voloshin A.F.
On one general scheme of the method of sequential analysis and elimination of variants. - Kibernetika, 1978, No.5, pp. 98-105.
- [14] Mikhalevich V.S., Shor N.Z.
Numerical solutions of multivariant problems by the method of sequential analysis of variants. - Nauchno-metodicheskie materialy ekonomiko-matematicheskogo seminara. LEMI AN SSSR. M., 1962, vyp.1., pp. 15-42.
- [15] Mikhalevich V.S.
Sequential optimization algorithms and their employment. Kibernetika, 1965, No.1. pp.45-55, No.2, pp.85-89.
- [16] Benders J.F.
Partitioning procedures for solving mixed variables programming problems. - Numer. Math., 1962, 4, No.3, pp.238-252.
- [17] Computational methods of the choice of optimal design solutions. /ed. V.S. Mikhalevich/. Kiev: Naukova dumka. 1977, 178p.

- [18] Kajurov V.Ju., Kuksa A.I.
Method of sequential analysis of variants in discrete problem of allocation of limited resources. Kiev: IK AN USSR, 1971. 13 p.
- [19] Fisher M.L.
Optimal solution of scheduling problems using Lagrange multipliers. Operations Research, 1973, 21, No.5, pp. 1114-1127; Lect. Notes Econ. and Math.Syst. 1973, 86. pp. 294-318.
- [20] Kuksa A.I., Laptin Ju.P.
Dynamic programming in network scheduling problem. Kibernetika, 1978, No.1, pp. 111-114.
- [21] Kuksa A.I.
Estimates of duration of shortest schedules in network planning problem with nonstorable resources. - In: Teorija optimal'nykh reshenij. - Kiev: IK AN USSR, 1975.
- [22] Nasekin V.Ja.
One approach to distribution of machines among jobs. In: Management information systems. Kiev: IK AN USSR, 1979, pp. 39-40.
- [23] Trubin V.A.
On method of solving special integer linear programming problems. - Doklady AN SSSR, 1969, 189, No.5, pp.952-954.
- [24] Trubin V.A., Vojtishin Ju.V.
An algorithm for solving set partitioning problem. Kibernetika, 1981, No.2, p.136.
- [25] Nasekin V.Ja.
Method for solving monotonous problems of integer programming. - Kibernetika, 1979, No.5, pp. 106-109.

PROGRAM-PACKAGE NLOP-1 FOR NONLINEAR
CONSTRAINED OPTIMIZATION

G.Christov, M.Ivanchev, R.Kaltinska,
Z.Karamiteva, W.Kjuchukova, J.Mitev

(Sofia, Bulgaria)

1. DESTINATION

The Program-package NLOP-1 is created in Operations Research Department at the Centre of Mathematics and Mechanics of the Bulgarian Academy of Sciences. It is intended for usage on ES computing system. The package contains programs solving the following nonlinear optimization problems:

- fractional programming problems;
- general and special quadratic programming problems;
- mixed integer programming problems;
- mixed integer programming problems with grouped variables;
- quadratic assignment problems;
- knapsack problem;
- travelling salesman problem.

2. PROBLEMS AND ALGORITHMS

1. General fractional programming problem

The fractional programming problem is stated as follows

$$\sup_{x \in M} \left\{ H(x) = \frac{p_0 + \sum_{j=1}^n p_j x_j}{q_0 + \sum_{j=1}^n q_j x_j} = \frac{P(x)}{Q(x)} \right\}$$

$$M: \begin{cases} \sum_{j=1}^n a_{ij} x_j^{(*)} b_i, & i=1, 2, \dots, m \\ x_j \geq 0 & j=1, 2, \dots, n \end{cases}$$

The symbol $(*)$ means an arbitrary combination of $\{\leq, =, \geq\}$.

The program for this problem, named HIPER, is a realization of the method described in [1]. This method has the advantage that no restrictions for the set M and for the behavior of the objective function $H(x)$ over M are prescribed. Usually the other methods require that $Q(x) > 0$ for every $x \in M$. The latter condition is rather difficult to check. On the other hand, there are problems which have solution even when $Q(x) = 0$ for some $x \in M$.

The base of the method is a simplex procedure (multiplicative version of revised simplex method) with special choice of initial basic feasible solution, of movement from a given feasible solution to another one and with a special optimality criterion for the fractional programming problem.

By means of this method the program HIPER investigates fully the problem. As a result the program finds an optimal solution (if there is any) or a suboptimal and an asymptotic one (see [1]) in the case when $H(x)$ is bounded over M , but does not achieve its supremum. Otherwise HIPER finds out the reason for the unsolvability ($M = \emptyset$ or $H(x)$ is unbounded over M) and gives an appropriate message. As a first step HIPER checks whether $H(x)$ is a constant or a linear function over M , then it gives an appropriate message and in the linear case the problem is solved as a linear programming problem. This allows HIPER to be used for solving of LP problems.

2. Quadratic programming problems

The following programs for solving quadratic problems are included in the package:

- program QUADRO for the general quadratic problem;
- programs QUADP, QUADPT and QUADT for some special classes of quadratic problems.

2.1 The program QUADRO solves the following problem:

$$\min \{Q(x) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n q_{ij} x_i x_j + \sum_{j=1}^n c_j x_j\}$$

subject to

$$\sum_{j=1}^n d_{ij} x_j (*) b_i, \quad i=1, 2, \dots, m$$

$$x_j \geq 0, \quad j=1, 2, \dots, n$$

The matrix $(q_{ij})_n$ should be a symmetric and positive semi-definite one. QUADRO is a realisation of Cottle's method [2]. This is an exact method based on a special approach for using Kuhn-Tucker's necessary conditions. The method is close to the Lemke's method which could be used to quadratic problems. The advantages of Cottle's method are: comparatively small dimensions of the simplex tableaux and small number of computations needed. The method is recommended mainly for problems with inequality constraints.

2.2 The programs QUADP and QUADPT solve the problem

$$\min \{Q(x) = \sum_{j=1}^n c_j x_j^2\}$$

under the restrictions

$$\sum_{j=1}^n x_j = q$$

$$a_j \leq x_j \leq b_j, \quad j=1, 2, \dots, n.$$

The program QUADP is specialized for the case when $c_j=1$, $j=1, 2, \dots, n$. Both programs are realisations of an algorithm proposed by G. Christov but unpublished yet. A special approach based on Kuhn-Tucker's necessary conditions is used. The algorithms are rather fast and need comparatively small storage for calculations.

2.3 The program QUADT solves the problem:

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}^2$$

under transport-type constraints

$$\sum_{j=1}^n x_{ij} = a_i, \quad i=1, 2, \dots, m.$$

$$\sum_{i=1}^m x_{ij} = b_j, \quad j=1, 2, \dots, n.$$

$$x_{ij} \geq 0, \quad i=1, 2, \dots, m, \quad j=1, 2, \dots, n.$$

The algorithm realised in the program is proposed by G. Christov as well. It has the same time and storage characteristics as the algorithms of 2.2.

3. Mixed integer programming problem

The mixed integer programming problem is stated as follows:

$$\min(\max) \left\{ \sum_{j=1}^{n_1} c'_j x_j + \sum_{j=1}^{n_2} c''_j y_j \right\}$$

by conditions

$$Ax + By (*) b$$

$$x \geq 0 - \text{integer}$$

$$y \geq 0$$

The program named MILC is a realization of an improved branch and bound-type algorithm described in [3,4]. It is characterized by: a compact record of information - the program needs storage only for the initial simplex tableau, preliminary cutting of unperspective branches without solving relaxed problems etc.

Problems with up to 100 integer variables and 50-60 constraints have been solved. MILC can be used for solving integer problems ($u_2=0$) or for linear problems ($u_1=0$).

4. Mixed-integer programming problem with grouped variables

The program named GROUP solves the following discrete problem:

$$\min(\max) \{ (c_1, x) + (c_2, y) + \dots + (c_k, z) \}$$

under the restrictions

$$Ax + By + \dots + Cz (*) b$$

$$x \in \{x^1, x^2, \dots, x^p\}$$

$$y \in \{y^1, y^2, \dots, y^q\}$$

$$z \in \{z^1, z^2, \dots, z^r\}$$

where x, y, \dots, z are vectors with (in general) different dimensions; x^1, x^2, \dots, x^p are given values of these vectors. Many practical problems can be adequately described by this model, especially when optimization among fixed variants must be made.

The program is a realization of the branch and bound algorithm ([4]) modified for this problem. Some of the variables could be continuous.

5. Quadratic assignment problem

Three programs QUAP, HILCO, QUAPA are concerned with the problem:

$$\min \left\{ \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n c_{ik} b_{jk} x_{ij} x_{kl} + \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_{ij} \right\}$$

$$\sum_{i=1}^n x_{ij} = 1, \quad j=1, 2, \dots, n$$

$$\sum_{j=1}^n x_{ij} = 1, \quad i=1, 2, \dots, n$$

$$x_{ij} \in \{0, 1\}, \quad i=1, 2, \dots, n, \quad j=1, 2, \dots, n.$$

5.1 The program QUAP is a realization of the exact method described in [5]. It is a modification of branch and bound-type algorithm with LIFO (last in first out) rule of branching. It gives comparatively fast the first feasible solution. As a result the program presents a list of feasible solutions the last of which is optimal. Problems with $n \leq 30$ have been successfully solved with QUAP.

5.2 The program HILCO is a realization of the well-known Hiller and Connors method ([6]), and the method realized in QUAPA is described in [7]. Both give an approximative solution. QUAPA is convenient for solving large dimensional problems because it is fast, there is no losses of accuracy and a small computer storage for calculations is needed. The average deviation between the optimal solution and those found by HILCO and QUAPA is less than 10%.

6. Knapsack problem

The well-known knapsack problem

$$\max(\min) \{(c, x)\}$$

under the conditions

$$(a, x) (*) b$$

$$x \in \{0, 1\}$$

is solved by the program named KNAPS. It is realization of two methods: a dynamic programming method and a branch and bound-type algorithm. The program chooses one of these methods in accordance with the problem's dimensions and the preliminary given computing time T . If T is large enough the problem is solved precisely by the dynamic programming method. Otherwise it is solved by the second method and at least one feasible solution is obtained.

7. Travelling salesman problem

We use the following description of the travelling salesman problem:

$$\min \sum_{i=0}^n \sum_{j=0}^n c_{ij} x_{ij}$$

by conditions

$$\sum_{i=0}^n x_{ij} = 1, \quad j=0, 1, \dots, n$$

$$\sum_{j=0}^n x_{ij} = 1, \quad i=0, 1, \dots, n$$

$$u_i - v_j + n x_{ij} \leq n-1, \quad i, j=1, 2, \dots, n, \quad i \neq j$$

$$x_{ij} \in \{0, 1\}, \quad i, j=0, 1, \dots, n$$

$$u_i \geq 0, \quad i=1, 2, \dots, n.$$

The program named TSP is intended for solving problems with symmetric distance matrix as well as with an asymmetric one. The program is a modification of the well-known algorithm by Little, Murty et al which is of branch and bound-type. Since small storage for calculations is used, comparatively large problem could be solved. Problems with up to 70 towns have been solved with TSP.

3. ORGANIZATION OF THE PACKAGE

Each of the listed programs is formed as a phase and can be used independently. The package organization of the program is a bit formal but their structure and functions are unified.

Each program (phase) fulfils the following functions:

1/ Data loading from arbitrary device (card reader, magnetic tape, disk) with syntactical control for all the data and error's message. If errors are found the program stops the performance.

2/ Dynamic storage distribution depending on the problem's dimensions. If the storage is not enough the program stops and an appropriate message is given.

3/ Solving of the problem.

4/ Print of the initial data and of the results. There are several print's levels and therefore various volume output information can be obtained. By means of these print's levels (controled by some parameters) one can trace the calculation process.

Each program (phase) consists of a main program, a main subroutine, a subroutine ADDRES for the dynamic storage distribution, and some auxiliary subroutines. All these program blocks are written in FORTRAN except for ADDRES which is in ASSEMBLER. They have the following functions:

1. Main program:

- reads problem's dimensions;
- calls ADDRES for dynamic storage distribution; stops the performance in case of insufficient storage;
- calls the main subroutine.

2. Main subroutine:

- reads the initial data and makes syntactical control for all the data; gives a message about any found error; stops the performance and gives message when errors are discovered;

- solves the problem;
- prints the input-output information.

3. Auxiliary subroutines provide service for the main subroutine.

4. APPLICATIONS

The package is implemented in many computer centers by Bulgarian Central Program's Library. One of the mostly used applications of the package is this in the computer aided design of engineering systems.

R E F E R E N C E S

- [1] Christov, G.,
Hyperbolic Optimization Problem, Paper presented at IX
International Symposium on Mathematical Programming,
Budapest, August 25-27, 1976.
- [2] Cottle R.W.,
The principal pivoting method of quadratic programming,
Math. Deas. Sci. Part I, Providence R.I., Amer. Math.
Soc., 1968.
- [3] Митев Й.Г.,
Применение метода ветвей и границ к некоторым задачам
дискретного программирования, ЖВМ и МФ, №2, 1977, стр.518-522
- [4] Митев Й.Г.,
Об одном штрафе в дискретном программировании, ЖВМ и МФ,
№6, 1977, стр. 1597-1598.
- [5] Иванчев М.Д.,
Об одном классе квадратичных задач, ЖМВ и МФ, №5, 1976,
стр. 1348-1353.
- [6] Hillier S., Connors M.,
Quadratic assignment problem algorithms and the location
of indivisible facilities, Manag. Sci. 13, 1966, N.1.
- [7] Калтинска Р., Дренчева Р.,
Алгоритми за решаване на квадратична задача за назначения-
та, сб. Мат. и мат. образование, 1977.

PROGRAM PACKAGE NLOP-2 FOR NONLINEAR OPTIMIZATION

G.Christov, A.Donchev, M.Ivanchev, R.Kaltinska
Z.Karamiteva, Z.Nedeva, B.Parakozova, W.Vellov

(Sofia, Bulgaria)

I. DESTINATION

The program package NLOP-2 is prepared in Operations Research Department at the Centre of Mathematics and Mechanics at the Bulgarian Academy of Sciences. It is intended for usage on ES computing system. The package contains programs solving the following nonlinear optimization problems:

- one dimensional minimization (11 subroutines);
- unconstrained minimization (18 subroutines);
- constrained minimization (2 subroutines);
- minimization by special type polyhedra constraints (2 subroutines);
- optimal control problems for linear systems with quadratic costs.

II. PROGRAMS AND ALGORITHMS

1. One dimensional minimization

The subroutines solve the problem

$$\min_{\alpha \geq 0} F(x^0 + \alpha d)$$

where $F: R^n \rightarrow R$ and $x^0 \in R^n$, $d \in R^n$ are given.

The methods used are:

- the binary search method ([1], p.18);
- the golden search method (two variants) ([1]), pp.32-35, [2], 49-50);
- the quadratic interpolation method ([1], pp. 44-47);
- a combination of the golden search and the quadratic interpolation methods ([1], p.44-47);
- a combination of the quadratic and the cubic interpolation methods (unpublished);
- the tangents method ([1], p.38);
- the one-sided step search method ([2], p.56);
- the cubic interpolation method ([3]);
- the binary search method using the gradient;
- a combination of the quadratic interpolation and the one-sided search methods ([1]).

Only the last 7 of the methods mentioned above use the gradient of the function considered.

A subroutine for a rough estimation (interval) of extremal point is included. It could be called as a service subroutine from the other subroutines. Depending on the demands of users it could be used separately.

2. Unconstrained minimization

The subroutines solve the problem:

$$\min_{x \in R^n} F(x)$$

Such problems arise often in the practice. They are frequently used by the methods for constraint function minimization. That is why the subroutines of that group are principal in the package with respect to diversity and applicability. The methods included are of proven efficiency and belong to the classes of nongradient methods, variable metric methods (quasi-Newton) and gradient methods. The following is a complete list of the methods used:

- the coordinate descent method;
- Powell's method [4];
- Powell-Zangvill's method [5];
- Rosenbrock's method ([6], p.173);
- Hooke-Jeeves method [6];
- Davis-Swann's method [6];
- Nelder-Mead's method ([6], p.363);
- steepest descent method [1];
- Fletcher-Reeves' method [7];
- Pollac-Ribiere's method;
- Newton's method [6];
- Woolf-Broyden-Davidon's method [8];
- Davidon-Fletcher-Powell's method [9];
- Fletcher's method [10].

Each of the Pollac-Ribiere's and Woolf-Broyden-Davidon's methods are realized in 3 variants (Subroutines). The variants differ in the computation of the gradient (analytically or numerically) and in the realizations of the algorithms.

3. Constrained minimization

The subroutines solve the problem:

$$\min_{x \in Q} F(x)$$

where $Q = \{x \in R^n \mid f_i(x) \leq 0, h_j(x) = 0, i=1, 2, \dots, m, j=1, 2, \dots, m_1\}$

Two methods are realized: a method using internal and external penalty functions and a method using modified Lagrange function.

4. Minimization by special type polyhedra constrains.

The subroutines solve the problem:

$$\min_{x \in Q} F(x)$$

where $Q \subset R^n$ is parallelepiped i.e. $Q = \{x \in R^n | a \leq x \leq b\}$ or polyhedra given by constraints

$$\begin{aligned} c_1 &\leq x_1 \\ x_1 &\leq x_2 - c_2 \\ &\dots \\ x_{n-1} &\leq x_n - c_n \\ x_n &\leq c_{n+1} \end{aligned}$$

where

$$c_{n+1} - c_n \geq \sum_{i=2}^n c_i, \quad c_i \geq 0, \quad i=1, 2, \dots, n+1$$

The method realized is the conditional gradient method [2].

5. Optimal control of linear systems with quadratic costs.

The following three problems are considered.

5.1. Optimal control problem for the system

$$(2) \quad \dot{x} = A(t)x + B(t)u, \quad x(t_0) = x^0$$

and performance index

$$(3) \quad I(u) = x^T(T_1)Gx(T_1) + \int_{T^0}^{T_1} [u^T(t)Q(t)u(t) + x^T(t)R(t)x(t)] dt$$

where $x \in R^n$, $u \in R^n$ and $A(t), B(t), Q(t), R(t), G$ are matrices with given properties, $[T_0, T_1]$ is fixed time interval. A method of difference approximation [12] is realized. The problem is reduced to a discrete optimal problem and the quadratic programming problem obtained is solved by a gradient method.

5.2. Optimal control problem of linear system (2) and for the cost functional

$$I(u) = x^T(T_1) G x(T_1) + \int_{T_0}^{T_1} u^T(t) Q(t) u(t) dt$$

A gradient method in which the optimal control is explicitly expressed [13] is realized in the program.

5.3. Optimal control problem for the system

$$\frac{d^n x_1(t)}{dt^n} + a_1 \frac{d^{n-1} x_1(t)}{dt^{n-1}} + \dots + a_n x_1(t) = u(t)$$

$$\frac{d^{i-1} x(T_0)}{dt^{i-1}} = x_i^0, \quad i=1, 2, \dots, n$$

(it is assumed that the characteristic polynomial of the equation does not have multiple roots). The performance index is the following

$$I(u) = \int_{T_0}^{T_1} [x^T(t) R x(t) + Q u^2(t)] dt$$

where R and Q are permanent matrices and

$$x(t) = (x_1(t), \frac{dx_1(t)}{dt}, \dots, \frac{d^{n-1} x_1(t)}{dt^{n-1}})$$

The classical Ritz method [14] is realized. For the unconstrained function minimization an arbitrary subroutine of 2. can be used.

III. PACKAGE STRUCTURE AND ORGANIZATION

The package is a collection of Fortran subroutines. There are 36 basic subroutines intended for solving the problems of section II and 16 auxiliary subroutines performing some standard operations as gradient norm calculation, inner product

calculation etc. These subroutines could be used in various programs for solving optimization problems.

All subroutines use the main storage only and are free of input-output statements. The only exception are the subroutines for printing the results.

The main goals of the package organization are:

- to be convenient for the users;
- to guarantee the solving of "good" problems (i.e. problems satisfying the requirements of algorithms)
- to provide a possibility for users' intervention in the computational process in order "non-good" problems to be solved as well;
- to provide a possibility for estimation of the efficiency of the method used and for adjustment of their parameters on the base of statistics gathered in the runs of problems with different characteristics.

These goals are met with the following organization:

1. The calls of the subroutines from the same group are equal, i.e. a standard set of formal parameters is used. One-dimensional minimization subroutines are formal parameters of unconstrained minimization subroutines and these, on the other hand, are formal parameters of constrained minimization and optimal control subroutines. This allows every subroutine to be changed without interrupting the computing process.

2. All parameters connected only with the chosen algorithm (as test accuracies, initial steps, "computer zero" etc.) are not fixed and could be changed by the users. This turns out to be very important when "non-good" problems are solved and allows to find out experimentally "best" values for some parameters. The parameters' default values are given in the package documentation for users' convenience.

3. A lot of parameters which provide information during a program run are introduced: e.g. control parameters, output control parameters, statistics parameters. They could be used for a control of the computing process and/or for tracing the run.

4. The output has several levels with different volume of information. By means of the output control parameters one could determine: the output level, the number of iterations to be skipped on the output file, the number of iterations between two successive outputs, etc.

5. The subroutines' call is simplified by using two common areas (for real and integer data). Each one of the groups of subroutines given in the section I gets the parameters needed from fixed fields of these common areas. Another advantage of such organization is that the control and the tracing could be done simultaneously when using the subroutines at different levels of calling.

IV. APPLICATIONS

The package is available for use at the Bulgarian Central Program's Library which has implemented it in various computer centers. One of the mostly used applications at the package is this in the Computer aided design of engineering Systems.

R E F E R E N C E S

- [1] Васильев Ф.П.,
Лекции по методам решения экспериментальных задач, Изд.
МГУ, 1974.
- [2] Полак Э.,
Численные методы оптимизации, М., Изд. МИР, 1974.
- [3] Fletcher R., Reeves C.M.,
Function minimization by conjugate gradients, Comput.J.,
Vol. 7, 1964, p. 147.
- [4] Powell M.J.,
An efficient method of finding the minimum of a function
of several variables without calculating derivatives,
Comput. J., Vol. 7, 1964, p. 155.
- [5] Zangwill W.I.,
Minimizing of function without calculating derivatives,
Comput. J., Vol.10, 1967, p. 293.
- [6] Химмельблау Д.,
Прикладное нелинейное программирование, Изд. Мир, 1975.
- [7] Карманов В.Г.,
Математическое программирование, М. Изд. Наука, 1975.
- [8] Broyden G.G.,
Quasi-Newton methods and their application to function
minimization, Math. Computation, Vol. 21, 1967, pp 368-381
- [9] Fletcher R., Powell M.,
A rapidly convergent descent method for minimization,
Comput. J., Vol 6, 1963, pp. 163-168
- [10] Flether R.,
A new approach to variable metric algorithms, Comput. J.,
N 3, Vol. 13, 1970.

- [11] Голштейн Е.Г., Третьяков Н.В.,
Градиентный метод минимизации выпуклого программирования,
связанный с модифицированными множителями Лагранжа, Эконо-
мика и математические методы, 1975, том XI, вып. 4.
- [12] Будаков Б.М., Васильев Ф.П.,
Некоторые вычислительные аспекты задач оптимального управ-
ления, Изд. МГУ, 1975.
- [13] Ли Э.В., Маркус Л.,
Основы теории оптимального управления, Изд. Наука, 1972.
- [14] Атанас М., Фалб П.,
Оптимальное управление, М., Машиностроение, 1968.

ON A SOFTWARE PACKAGE FOR SOLVING NONLINEAR PROGRAMMING PROBLEMS

C. Richter

(Dresden, GDR)

1. INTRODUCTION

Nonlinear programming problems often occur in economics, technics and natural sciences. For solving such problems a program package of nonlinear programming is presented. It contains a collection of effective algorithms for solving constrained problems of medium size. The optimization methods have been implemented by members of the research department "Numerical methods of optimization" of the section mathematics at the TU Dresden.*)

In the package NLOPT the following principles are realized:

- All optimization algorithms operate with a standard form of computational descriptions of the programming problem. Thus a given problem can be handled by various algorithms without additional changes in the implementation.
- The program package possesses block structure and sub-problems of the same kind are solved by auxiliary moduls.
- The optimization algorithms are implemented as subroutines in FORTRAN (tested at BESM-6 and EC 1022). The information available at the lineprinter is controllable by the user. The output-listings are standardized for all optimization routines.

The presented program package has been developed on the basic of well-known results from optimization-theory, own theoretical investigations and modifications of the algorithms

*) The members were (1979): C. Grossmann, A. Hahnwald-Busch, W.Koch, G.Langensiepen, C.Richter, K.Schönefeld, G.Terno

stimulated by some users in practice.

The package contains

- cutting-plane-methods
- shifted penalty methods
- generalized reduced gradient methods
- constrained Newton-type methods operating with second order approximation of the Lagrangian and first order approximations of the constraints.

To guarantee an effective application it was necessary to make demands on the program package:

a) The different treatment of linear and nonlinear constraints

The advantage of this consists in the simple description of linear terms in the problem only by the coefficients.

Furthermore the specific character of linear constraints can be taken into consideration to get more efficient algorithms.

b) Numerical approximation of the derivatives by function values only

In most of the problems the derivatives are not explicitly available or its analytical representation is practically impossible. On the other hand, if derivatives are available, they can be directly used instead of approximations.

c) Forcing feasibility of iteration points in certain algorithms

In some applications the feasibility of iteration points plays an essential role. In this case outer-point-algorithms cannot be applied.

2. GENERAL STRUCTURE OF THE PACKAGE

For solving a nonlinear programming problem by an algorithm contained in the package NLOPT the following standard description has to be used:

Let be $M1$, $M2$, $L1$, $L2$ and N nonnegative integers defining the dimension of the problem (number of constraints and numbers of variables).

Then we consider

$$f_{M1+1}(x) + A_{M2+1, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} \rightarrow \min!$$

subject to

$$\left. \begin{array}{ll} f_i(x) = 0 & i=1, \dots, L1 \\ f_i(x) \leq 0 & i=L1+1, \dots, M1 \\ A_{j, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} = 0 & j=1, \dots, L2 \\ A_{j, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} \leq 0 & j=L2+1, \dots, M2 \end{array} \right\} \quad (P)$$

$$US_k \leq x_k \leq OS_k$$

Thereby A denotes a $(M2+1, N+1)$ -matrix and $A_{j, \cdot}$ the j -th row of A . Furthermore US_k and OS_k are given lower and upper bounds for the k -th variable x_k and $f_i: R^n \rightarrow R^1$ ($i=1, \dots, M1+1$) denote the non-linear functions describing the problem.

To short our notation we set

$$z(x) = f_{M1+1}(x) + A_{M2+1, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} \quad \text{for any } x \in R^N$$

and define

$$D = \{x \in R^N : US_k \leq x_k \leq OS_k, A_{j, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} = 0, j=1, \dots, L2, \\ A_{j, \cdot} \begin{bmatrix} 1 \\ x \end{bmatrix} \leq 0, j=L2+1, \dots, M2\},$$

$$G = \{x \in D : f_i(x) = 0, i=1, \dots, L1, f_i(x) \leq 0, i=L1+1, \dots, M1\}.$$

Thus the problem (P) is equivalent to

$$\begin{array}{ll} z(x) = \min! & \\ \text{subject to } x \in G. & (P') \end{array}$$

The set D is given by the linear constraints of the non-linear programming problem (P) . The linear part of the problem

(P) and the dimension are to store in an array PF in a well defined manner (see [1]). Additional, some informations to control the algorithm are contained in PF . The nonlinear functions f_i are summarized in one subroutine F in the following way:

```
SUBROUTINE F(I,X,Y,IE)
DIMENSION X(1)
```

```
Y =
IE=
RETURN
END
```

Thereby for given parameters I and X the value Y is defined by $Y=f_I(X)$. The output parameter IE indicates the computability of Y . If $IE < 0$ then $f_I(X)$ is not defined. Similar to the functions f_i the related derivatives ∇f_i , $\nabla^2 f_i$ can be implemented in subroutines FD and $F2D$ respectively. If the derivatives are not available then approximations are automatically used in the package NLOPT.

Next we give a review of the moduls of the program package.

Optimization routines:

```
SE 1   - cutting plane method
GRG    - generalized reduced gradient method
MSB    - shifted penalty method (augmented Lagrangians)
WILSON }
NTYP   } - methods with local quadratic approximations
SSV1   - stochastical search method
```

Organization routines

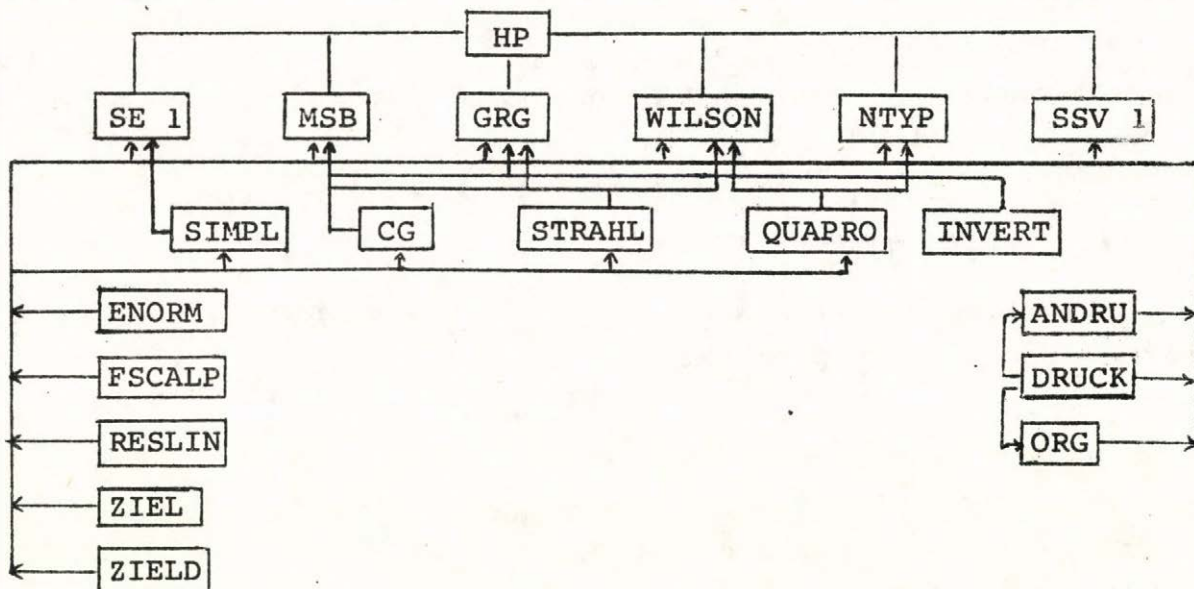
```
HP      - control program
ANDRU   - organization routines at the beginning of iterations
DRUCK   - routine for printing
ORG     - organization of the iteration cycles and test of
          standard stopping criterions
```


General subroutines

QUAPRO - BEALE's quadratic programming method
SIMPL - dual linear programming with upper bound techniques
CG - minimizing without constraints
(conjugate direction method, resp. updating technique)
STRAHL - one dimensional minimization
(golden section and interpolation)
INVERT - inversion of a matrix

Furthermore subroutines performing linear algebra problems and specific operations needed in some optimization algorithms are contained in NLOPT. The subroutines QUAPRO, SIMPL and CG can be directly used for related optimization problems (quadratic, linear or unconstrained).

The interaction between the moduls of program package can be characterized by the following scheme



3. MATHEMATICAL PRINCIPLES OF THE IMPLEMENTED OPTIMIZATION ALGORITHMS

In this chapter we describe the essential steps in the k -th iteration cycle of the methods included in the package NLOPT.

Cutting plane method (SE 1)

The iteration point x^k is determined by SIMPL as a solution of the problem

$$z(x) \rightarrow \min! \quad \text{subject to} \quad x \in P_k$$

$$\text{Set } P_{k+1} = \{x \in P_k : f_{i_k}(x^k) + \nabla f_{i_k}(x^k)^T (x - x^k) \leq 0\}$$

$$\text{with } f_{i_k}(x^k) = \max_{1 \leq i \leq M1} f_i(x^k).$$

Remarks

- the objective function is assumed to be linear, otherwise the problem is transformed such that the resulting objective has this property.
- SE1 realizes KELLEY's method
- P_0 is a given or automatically generated polyhedral set with $G \subset P_0 \subset D$.
- nonbinding constraints are dropped
- only convex problems can be handled by SE1 and the iterates x^k are not feasible, i.e. $x^k \notin G$.

Shifted penalty method (MSB)

The given optimization problem is transformed into a sequence of auxiliary problems

$$E_k(x) \rightarrow \min! \quad \text{subject to} \quad x \in M_k \tag{3.1}$$

Thereby, M_k , $k=1,2,\dots$ are open subsets of R^N and solutions x^k of (3.1) are known to be inside of M_k and can be determined by the subroutine CG. As the function E_k we use

$$E_k(x) = z(x) - \sum_{i \in I_B} \frac{1}{p_i^k (f_i(x) + u_i^k)} + \sum_{i \in I_S} p_i^k \max^2 \{0, f_i(x) + u_i^k\}$$

with given index sets I_B and I_S . The linear constraints of (P) can be handled in E_k in the same manner as the nonlinear ones

or can be preserved in (3.1) as additional constraints. In the later case NTYP or WILSON is used to solve (3.1).

Generalized reduced gradient method (GRG)

An appropriate feasible descent direction is determined in the following way:

At the point x^k two sets $\tilde{Z}(k)$ and $\hat{Z}(k)$ are defined. The set $\tilde{Z}(k)$ contains the indexes of maximal $M1+M2+2N$ inactive constraints $f_i(x)$ (including linear and nonlinear constraints, lower and upper bounds), the set $\hat{Z}(k)$ is formed by the complement (N elements). The gradients of the constraints $f_i(x)$ ($i \in \tilde{Z}(k)$) in the point x^k are used for constructing the matrix $\tilde{A}(x^k) = [f'_i(x^k)]_{i \in \tilde{Z}(k)}$.

By means of this matrix and the gradient of the objective function $z(x)$ the vectors v^k , y^k and s^k are defined due to

$$v^k = -(\tilde{A}(x^k)^{-1})^T z'(x^k)$$

$$y^k = \begin{cases} 0 & \text{if } i \in I_{\epsilon_k}^k = \{i: -\epsilon_k \leq f_i(x^k) \leq 0 \wedge v_i^k > 0\} \\ v_i^k & \text{in the other case} \end{cases}$$

$$\text{and } s^k = \tilde{A}(x^k)^{-1} y^k.$$

Finally the one-dimensional search procedure STRAHL is applied to get an approximate solution α_k of

$$z(x^k + \alpha s^k) \rightarrow \min!$$

subject to $\alpha \geq 0$ and $x^k + \alpha s^k \in G$.

The new iterate is given in the form

$$x^{k+1} = x^k + \alpha_k s^k.$$

Remarks

- if the inverse of A is not computable, then it is tried to change the index set $Z(k)$. In the case, that this is not successful the information "DEGENERATION" is given,
- Equality constraints cannot handled with GRG.

Wilson - method (WILSON)

Some Kuhn-Tucker-point $w^k = (y^k, v^k)$ of the quadratic subproblem

$$\begin{aligned} z'(x^k)^T(x-x^k) + \frac{1}{2}(x-x^k)^T B_k(x-x^k) \rightarrow \min! \\ \text{subject to } f_i(x^k) + f_i'(x^k)^T(x-x^k) \leq 0 \quad (i=1, \dots, L1) \\ f_i(x^k) + f_i'(x^k)^T(x-x^k) \leq 0 \quad (i=L1+1, \dots, M1) \quad (3.2) \\ x \in D \end{aligned}$$

is calculated.

In the undamped version the point w^k forms the new iterate $z^{k+1} = (x^{k+1}, u^{k+1})$.

In the damped version the new iterate is calculated due to

$$z^{k+1} = z^k + \alpha_k (w^k - z^k)$$

Thereby the positive parameter α_k is gotten from the linear search procedure STRAHL applied to minimize the measure of non-optimality

$$F(z) = \|L_x(z)\|^2 + \|g(x)^+\|^2 + (-u^T g(x))^+, \quad z = (x, u)^T, \quad g(x) = -(f_1, \dots, f_{M1})^T$$

Convergence properties of this techniques are prescribed in [8]. In the problem (3.2) the matrix B_k is chosen in different ways by calling related subroutines:

1. The Hessian $L_{xx}(z^k)$ of the Lagrangian of (P)
2. The rank-1 matrix

$$B_k = 2 B_{k-1} - \frac{(B_{k-1} q^{k-1} - \frac{1}{2} s^{k-1})(B_{k-1} q^{k-1} - \frac{1}{2} s^{k-1})^T}{(B_{k-1} q^{k-1} - \frac{1}{2} s^{k-1})^T q^{k-1}}$$

where $q^{k-1} = L_x(x^k, u^k) - L_x(x^{k-1}, u^k)$

and $s^{k-1} = x^k - x^{k-1}$ is. Furthermore $B_0 = I$ is chosen. This

approximation of the Hessian of the Lagrangian is transformed from the equivalent unconstrained method, given by KLEINMICHEL [6].

3. Some consist approximation of the Hessian of the Lagrangian

$$B_k h_i r^i = L_x(z^k + h_i r^i) - L_x(z^k), \quad i=1, \dots, N,$$

where h_i the steplength of the discretization in the i -th direction r^i (f.e. $r^i = e^i$, e^i -ith- unit direction) is [9].

4. The matrix B_k is defined by the positive definite matrix I . In this case the Wilson- method turns to the method of linearization written in [3]. The subproblems are described by the objective function

$$z'(x^k)^T(x-x^k) + \frac{1}{2}(x-x^k)^T(x-x^k) \rightarrow \min!$$

subject to the constraints of the problem (3.2).

Modified method of Levitin an Poljak (NTYP)

The algorithm NTYP is a special implementation of a quadratic approximation method. It handled only linear constraints. Thus the objective function will be quadratically approximated. Using a modified Cholesky- decomposition the subproblems are convex. The parameters in the decomposition are controlled by the measure of nonoptimality.

Stochastical search method (SSV 1)

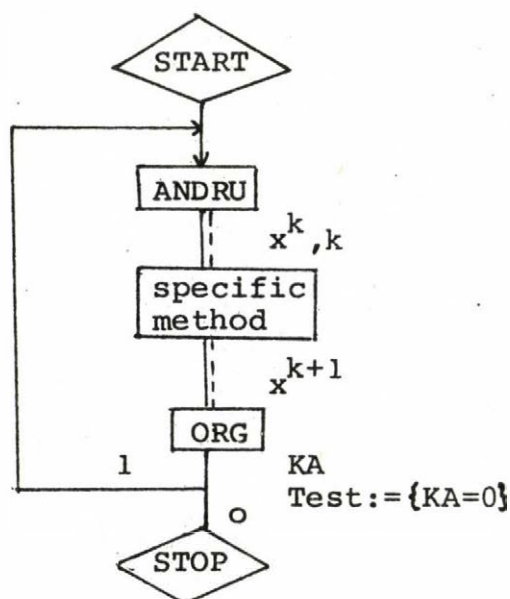
A sequence of test points is randomly generated. A new point is accepted as an iteration point if the objective function value decreases and the point is feasible. Using informations about the iteration points already generated the distribution is controlled. The SSV 1 algorithm is to prefer, if the problem functions are not differentiable or nonconvex. To

get higher accuracy a combination of SSV 1 with other methods can be proposed.

4. INPUT-OUTPUT-ROUTINES AND DATA FLOW

The unified description of the optimization problems given from the user and a simple readable output listing play an essential role for the applicability of the package. In NLOPT we realize the input and output of data by means of the routines ANDRU and DRUCK. Moreover all optimization problems are structured in the same way by organizing the iteration cycle with the routine ORG. Thus the computational effort for verifying stopping criterions, for the output organization and for the updating of data is nearly the same in each of the implemented optimization programs. This enables us to compare the various algorithms by means of the consumed computer time or of the number of function evaluations. On the other hand the user can easily understand the structured programs.

The following scheme shows the structure of the realized optimization algorithms.



The routine ANDRU defines the starting time and sets FWA=0 (number of function value evaluations) and $k=0$ (iteration index). Moreover, if the input datas are stored in a compact manner in the array PF then in the routine ANDRU these data are posed at the position needed in the sequel in the algorithm and some addition informations are filled in PF if necessary. The print of unified headlines and of the input datas is organized directly or by means of the routine DRUCK.

Now we consider the subroutine ORG. At the beginning of the k -th cycle the iteration point x^k and the value $z(x^k)$ of the objective function is known. The next iterative x^{k+1} will be defined by the specific method of the implemented algorithm. If this is not possible and the step can't be repeated in a modified manner such that x^{k+1} can be found then the iteration ends and the information that a subproblem is not solvable (KA=2) is transmitted to ORG. This can be modified to "empty feasible set" (KA=4) or "objective function unbounded" (KA=3) if one of this informations is available. In the subroutine ORG first the stopping criterions tested outside are checked. If no stop is detected then in ORG standard tests are performed.

In the package are realized the following stopping criterions:

- the problem can't be solved by the selected method (KA=-2)
- error in input datas (KA=-1)
- subproblem not solvable (KA=2)
- unbounded objective function (KA=3)
- empty feasible set (KA=4)
- iteration number exceeds its limit (KA=5)
- time consumption exceeds its limit (KA=6)
- $|z(x^{k+1}) - z(x^k)| \leq (10^{-2} + |z(x^{k+1})|) * EPS$ (KA=7)
- $\|x^{k+1} - x^k\| \leq (10^{-2} + \|x^{k+1}\|) * EPS$ (KA=8)
- a specific optimality test is fulfilled, $TEST \leq EPS$ (KA=9).

Moreover, additional tests can be performed out of the routine ORG and by setting the indicator KA greater or equal

10 the interruption is realized with the aid of ORG in the same manner.

To avoid undesired stops most of the tests can be excluded by informations in PF(10).

The output at the lineprinter is organized by the routines DRUCK and ANDRU. The information level can be controlled with the parameter IPRINT. Especially no print occurs if $IPRINT \leq 0$. An example for an output listing generated in the package NLOPT is given in table 1.

5. MAINPROGRAM

The optimization algorithms contained in the program package NLOPT are implemented in the form of separate subroutines. Thus the user can directly call our optimization routines from his own program to solve nonlinear programming problems as subproblems in connection with other computations. For the users interested only in solving optimization problems without additional operations we have developed a mainprogram to simplify the application of the algorithms contained in NLOPT.

The principle of the mainprogram consists in defining all parameters controlling the optimization routines as standard values. Thus the user only has to supply the datas describing his problem and the information which algorithm is wanted to apply.

If more informations in are available then the standard dates can be changed. This change is realized by several key words and related datas. For example, if the user wants the accuracy 10^{-5} , then two data cards are needed:

EPS

0.00001.

Now we give a short survey of the needed datas and possible key words controlling the mainprogram. We describe the version of the package NLOPT run at the computer EC 1022 using single precision (REAL*4).

After activating the mainprogram by //EXEC first the following 8 integers N,L1,L2,M1,M2,KG,IK,IS are necessary. Thereby N,L1,L2,M1 and M2 denote the problem dimension of (P). The value of KG characterizes the description of the objective function (KG=0 - nonlinear terms occur). With the aid of IK,IS a dense input of the array PF are to contained in the input stream.

For changing standard datas the following key words are preserved: X (starting point), EPS (final precision), EPST (temporary precision), IPRINT (parameter to control the output level), LTMAX (maximal time), ITMAX (maximal iteration number), IABBR (parameter to eliminate some stopping rules), KA (parameter for controlling the selected algorithms), OS (upper bounds), US (lower bounds), ST (type of bounds), OSI (one upper bound), USI (one lower bound), STI (one type of bounds), W and IV (addresses of the working arrays).

To select an algorithm the code words CG, MSB, NTYP, WILSON, GRG, SE1, SIMPL, SSV1 and QUAPRO are possible.

After finding one of this words the related algorithm is called. Then a new word will be decoded.

The mainprogram regularly stops if the code word EXIT is detected. By means of XALT or XNEU a parallel or sequential run of different algorithms can be organized. The status XALT is useful to compare several algorithms because the starting points are identical. After XNEU the solution obtained in the previous method is used as the starting point in the next algorithm.

Presenting a simple example we want to illustrate in table 1 how to realize an optimization and how to apply the main program.

6. SELECTED EXAMPLES OF PRACTICAL APPLICATIONS

The package NLOPT has been employed to solve problem arising in technical engineering. Here we present some problems handled by NLOPT and related questions of implementation


```

SUBROUTINE F(I,X,Y,IE,
DIMENSION X(1)
COMMON /FWA/J
IE=-1
J=J+1
Y=(10.*X(1)*X(1)+7.*X(2)*X(2))/(17.+3.*X(1)*X(1)+2.*X(2)*X(2))
RETURN
END

```

***** STRAF-BARRIERE-VERFAHREN (MODIFIZIERT)

PROBLEMPARAMETERI

N = 2 L1 = 0 M1 = 0 L2 = 0 M2 = 1 KG = 2

KOEFFIZIENTEN DER LINEAREN RESTRIKTIONEN:

** 0.10000000000E 01 -0.50000000000E 01 0.40000000000

ZIELFKT: 0.3093834E 00

*****10.11.77*****

X

VERLETZTE SCHR.

1 -.9999996E-01
2 -.9000000E 00

ITMAX = 10 IPRINT = 6 EPSANF =0.1000E-03 EPSEND =0.1000E-03

0E 01

** STARTPUNKT

NL - RESTRIKTIONEN - LIN

UNTERE - SCHRANKEN - OBERE

-.2099999E 01UNGL

-0.1000000E 01
-0.1000000E 01

0.1000000E 01
0.1000000E 01

Table 1

**** ERHALTENE LOESUNG NACH 10 SCHritten *****
ZIELFKT: 0.1224850E-01 NORMDIFF: 0.64144E-05 TEST: 0.10000

X

VERLETZTE SCHR.

NL - RESTRIKTION

1 0.1052477E 00
2 -.1184449E 00

Cont. Table 1.

***** ABRUCHURSACHE: MAXIMALE ITERATIONSZAHL *****

E-02 F=AUFRI 1131 TEILZEIT: 4 ZEIT: 46
EN - LIN NL-PARAMETER-LIN NL-SHIFT-LIN
-.1716614E-04UNGL 0.656E 03 0.444E-04

- Optimization of chemical reactor systems

In cooperation with M.Grauer from the TH"Carl Schorlemmer" Leuna-Merseburg we applied the program package to optimize the so called WILLIAMS-OTTO-PLANT. The problems obtained here are highly nonlinear in the objective function, which is not analytically available in the original form of the description. The related function values are defined by the solution of a system of nonlinear equations. It is possible to avoid the inner iteration process for solving this equations in each step by a modification of the original problem:

The equations are written in the form of equality constraints by introduction of additional variables.

This is an advantage. The disadvantage is given by the increasing of the number of variables and the number of the constraints. Additional the structure of the transformed problem is usually more complicated as the original problem, e.g. instead linear constraints the problem possesses linear and nonlinear constraints. The comparison of this two ways yields the following results

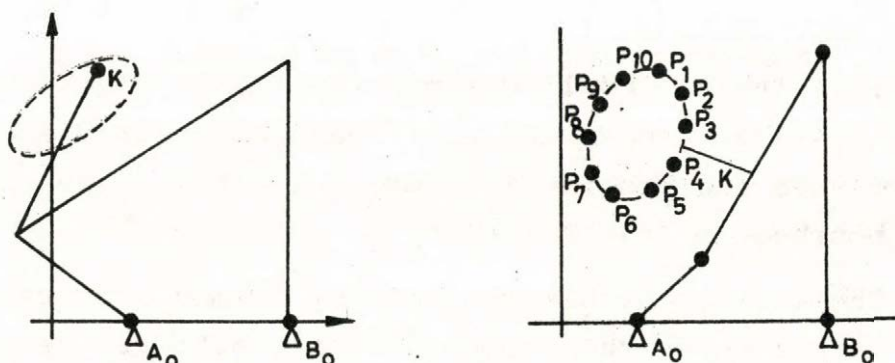
Execution time	MSB	GRG	WILSON	COMPLEX*	SSV 1
<hr/>					
Original problem					
(N=5 with lower and upper bounds)	26.46	7.36	6.48	11.61	52.83
<hr/>					
Transformed problem					
(N=22, L1=17, 5 lower and 5 upper bounds)	11.76	-	15.26	-	-

*) COMPLEX was implemented in Merseburg

For further details we refer to [5]

- Mechanical transmission system optimization

Given is a mechanical transmission system. The dimension of the system parts shall be calculated in such a way that a given point K is moved on a defined loop L . From this an approximation problem results which we solved by the minimization of the defect in certain points $P_i (i=1, \dots, 10)$. The constraints are lower and upper bounds for the length of the system parts and one constraint is necessary to guarantee the working of the transmission system.



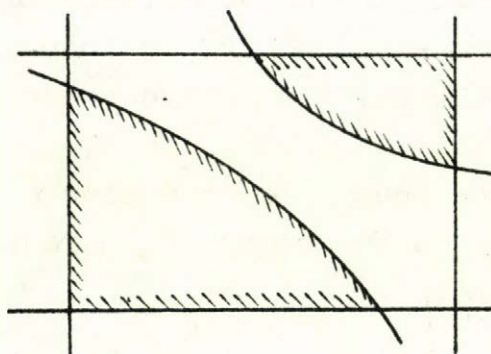
The problem was solved by the methods MSB, WILSON and SSV1. In the starting point the defect function has the value 762.9, in the solution points are the values 0.029, 0.00014 and 0.162 respectively.

A similar problem is the minimizing of mechanical charges under the constraints of a given tolerance related to the arcs between the parts of the system. Therefrom an optimization problem results, which contains 12 variables and 27 constraints. This problem was solved by the methods MSB, GRG, SSV1, SE1.

- Optimization in the microelectronic technology

The microelectronic-technology includes the tooling of quartz discs (partition, applying, of metallic structures a.s.o.). At this quadratic regression problems under quadratic and linear constraints occur. We receive extended quadratic programming problems, which are nonconvex. The feasible set is non-

connected. For example in a two-dimensional special case has the feasible set the following form:



To solve the nonlinear programming problems, which includes maximal 22 linear and nonlinear constraints and 4 variables it was used the program MSB, GRG and WILSON- the last two in a neighbourhood of the solution.

The various problems have many common properties. They can be interpreted as the elements of a homotopy. Therefore it is possible to use the solution of one problem as a starting point for a problem in its neighbourhood. In the given program package it is very simple to realize this "discrete continuation technique" by the following instructions:

1. Change the coefficients of the extended quadratic programming problem by the key words W,
the index of the first W-element,
the index of the first W-element, which contains
the coefficients,
the new coefficients;
2. Call the solution point of the last problem as a starting point of the new problem by the key word XNEU.

In a series of 6 examples with 4 variables and 8 constraints we get the following results:

Example	iteration number	execution time
1	12	30,6 s
2	4	10,4 s
3	3	7,6 s
4	3	7,5 s
5	3	7,8 s
6	4	11,0 s

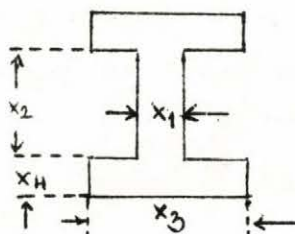
- Optimization of a steel girder

We consider the problem to determine the optimal area of a steel girder. The objective function is given by

$$z(x) = A = x_1 x_2 + 2x_3 x_4 \rightarrow \min!$$

subject to constraints, which are determined by the mechanical

$$\begin{aligned} \text{situation: } f_1(x, p_1, p_2) &\leq 0 \\ f_2(x, p_3) &\leq 0 \\ f_3(x) &\leq 0 \\ f_4(x) &\leq 0. \\ f_5(x) &= 0, \quad f_6(x) = 0 \end{aligned}$$



The parameters p_1, p_2, p_3 depend on the used material and the existing powers. Again it is possible to consider neighbouring situations as elements of a homotopy and in this way to reduce the execution time for their treatment. The program WILSON was used for solving a family of 30 such problems. For one parameter set it is demonstrated in chapter 7.

- Optimization of a water providing system

Given are m springs B_i ($i=1, \dots, m$) and n consumers M_j ($j=1, \dots, n$). For determining the optimal transportation volume $x_k = x_{ij}$ from B_i to M_j it is necessary to minimize the objective function

$$z(x) = \sum_{k=1}^{m \cdot n} ((\alpha_k x_k^{\beta_k} + \gamma_k x_k) + \lambda_k)$$

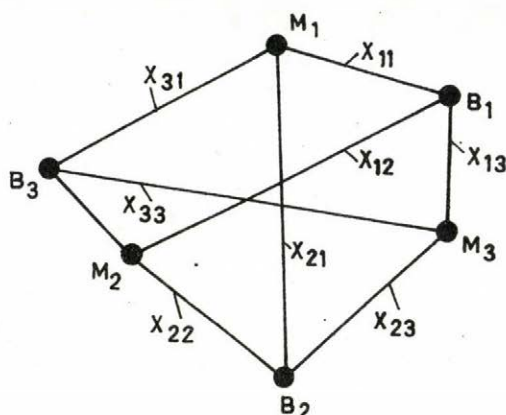
subject the constraints

$$\sum_{i=1}^m x_{ij} \leq U_{1j} \quad (j=1, \dots, n)$$

$$\sum_{j=1}^n x_{ij} = U_{2i} \quad (i=1, \dots, m)$$

$$0 \leq x_{ij} \leq R_{ij} \quad (i=1, \dots, m, j=1, \dots, n)$$

The structure of $z(x)$ results from the technological costs of the system - from the building costs and the working costs. The problem was written originally by MILASZEWSKI and ROMAN in [7]. For a problem of 12 variables and 11 restrictions they reached the objective function value 271.3. By using the subroutines NTYP it was possible to calculate a solution with the objective function value 25.2. The considered model was used - with more variables and more restrictions - for optimizing the water providing system in a district in the GDR.



7. NUMERICAL RESULTS

The program package was used for solving various test examples; so called 'real life test problems', described f.e. in chapter 6 and classical problems, f.e. given by COLVILLE [3] and by ROSEN/SUZUKI [10].

We summarize some selected test results now:

1. ROSEN/SUZUKI (Extended quadratic, $N=4$, $M1=3$):

	Cutting plane	Shifted pen.	Wilson	Wilson (Appr.)
It. numb.	73	19	5	7
It. time	6	6	2	3
Solution	-0.00549	0.00281	0.0	0.0
	0.99804	1.00060	1.0	1.0
	2.00096	1.99782	2.0	2.0
	-0.99925	-1.00249	-1.0	-1.0
Obj.funct.	-44.00006	-44.00001	-44.0	-44.0
Funct.ev.	365	3443	320	328

2. COLVILLE 1 (Linear constrained $N=5$, $M2=15$, $x_k=0$ ($k=1, \dots, 5$)):

	Shifted pen.	Wilson	NTYP
It. number	11	4	3
It. time	77	7	4
Solution	0.30000	0.30000	0.30000
	0.33358	0.33347	0.33358
	0.40000	0.40000	0.40000
	0.42859	0.42831	0.42858
	0.22397	0.22397	0.22370
Obj.funct.	-32.34869	-32.34868	-32.34866

3. Steel-girder-problem for a fixed parameter set:

We consider the problem to optimize the area of a steel girder for the parameters:

- $p_1 = 5600 \text{ Mpcm}$ - maximal bending moment
 $p_2 = 48 \text{ Mp}$ - maximal transverse load
 $p_3 = 1.5 \text{ Mp/cm}^2$ - feasible bending stress for steel sort
 SI 37.

The nonlinear programming problem has the form

$$z(x) = x_1 x_2 + 2x_3 x_4 = \min!$$

$$\text{subject to } f_1(x) = 50600 - (0.5x_1 x_2^2 + b x_4^2 + 3a^2 b) / (2a + 2x_4) \leq 0$$

$$f_2(x) = 24ab + 6c - 0.45(cx_2/6 + b x_4^2 / (3 + a^2 b)) \leq 0$$

$$f_3(x) = 0, 8 - x_1 \leq 0$$

$$f_4(x) = x_2 - 85.0 \leq 0$$

$$f_5(x) = 6x_4 - x_3 \leq 0$$

$$f_6(x) = x_3 - 20x_4 \leq 0$$

where $a = x_2 + x_4$, $b = x_3 x_4$ and $c = x_1 x_2^2$ is.

From the starting point $x^0 = (0.6, 75.0, 30.0, 1.75)^T$

with $z(x^0) = 150.0$

the solution $x^* = (0.80377, 74.99255, 29.95175, 1.49761)^T$

with $z(x^*) = 149.9004$

was reached after 16 iteration steps by MSB. The execution time was 35.4 s.

We remark, that the treatment of this problem is described incorrect in [4].

R E F E R E N C E S

- [1] Documentation of the program package NLOPT, Preprint TU Dresden 1980.
- [2] Colville, R.A.:
A comparative study of nonlinear programming. IBM Scientific Center Report No. 320-2949, 1968.
- [3] Danilin, Yu.N. and B.N. Psenicnij:
Numerical methods in extremal problems. Mir Publishes Moscow 1978.
- [4] Goliski, J. and S. Schönfeld:
"Vergleich verschiedener Optimierungsmethoden" in
"Beispiele zur rechnergestützten Optimierung von Konstruktionen", Working paper of Kdt, 1973.
- [5] Grauer, M., Gruhn, M. and Richter, C.:
Design and Application of the GRG-Algorithm for Process Optimization. Materials of the Conference "Contributions of Computers to the Development of Chemical Engineering and Industrial Chemistry", Paris 1978.
- [6] Kleinmichel, H.:
Quasi-Neston-Verfahren vom Rang-Eis-Typ zur Lösung unrestringierter Minimierungsprobleme, Numerische Mathematik, 38, 219-244 (1981).
- [7] Milaszewski, R. and M. Roman:
Matematyczny model wyboru zwoel wody.... Gaz, woda i technika sanitarna 53 (1979), 95-98.
- [8] Richter, C.:
Zur globalen Konvergenz des gedämpften Wilson-verfahrens, MOS, Ser.Opt. 10 (1979), 213-218.
- [9] Richter, C.:
Über Mehrschrittverfahren der nichtlinearen Optimierung, ZAMM 60(1980), 129-136.

- [10] Rosen, J. and S. Suzuki:
Construction of nonlinear programming test problems,
Comm. of the ACM 8 (1965), 113.

PROCEDURES FOR THE SOLUTION OF TRANSPORTATION TYPE PROBLEMS

G.Kéri, É.Komáromi, P.Sz.Turchányi

(Budapest, Hungary)

1. INTRODUCTION

In the Computer and Automation Institute of the Hungarian Academy of Sciences several researchers have been engaged in the expansion and programming of known algorithms to solve the transportation problem. As a result of this activity, computer programs of various type in FORTRAN language have been written for the CDC 3300 computer.

The most familiar solution methods of the transportation problem can be divided into two types: primal or dual in their main feature. The first ones are based on the adaptation of the primal simplex method (related to the general linear programming problem) to the transportation problem. On the other hand, the dual type methods are varieties of the primal-dual method specialized for the transportation problem, which can at the same time be considered also as special cases of methods developed for a more general flow problem.

Computer codes on the basis of primal type methods were written by Éva Komáromi and Gerzson Kéri. The generalization of the transportation problem for a piecewise linear, convex objective function was dealt by Piroska Sz. Turchányi, who has written a program based on the Dantzig-Wolfe decomposition and on the method of Williams.

2. MATHEMATICAL FORMULATION OF THE TRANSPORTATION PROBLEM

With one exception our programs have been written to solve the classical (i.e. Hitchcock's) transportation problem. This problem can be formulated in the following way:

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

subject to

$$\sum_{j=1}^n x_{ij} = a_i \quad (i=1, 2, \dots, m)$$

$$\sum_{i=1}^m x_{ij} = b_j \quad (j=1, 2, \dots, n)$$

$$x_{ij} \geq 0 \quad (i=1, 2, \dots, m; j=1, 2, \dots, n).$$

In case of the above-mentioned generalization for a piecewise linear, convex objective function, this latter will take, instead of the above, the form

$$\min \sum_{i=1}^m \sum_{j=1}^n \varphi_{ij}(x_{ij}),$$

while the constraints remain unchanged.

3. A MODIFIED VERSION OF THE PRIMAL SIMPLEX METHOD APPLIED FOR THE TRANSPORTATION PROBLEM

We have carried out a slight modification on Dantzig's adaptation of the simplex method to the transportation problem. This modification consists of another way of computation of the reduced costs $\delta_{ij} = z_{ij} - c_{ij}$, and was suggested by the recognition that at every change of basis the reduced costs are transformed in such a way that the same number is added to certain rows and at the same time subtracted from certain columns. This number is just the δ_{ij} belonging to the cell to enter the basis. The modification based on the above-mentioned recognition is applied in the subroutines RUN and TRNS.

Let H be an arbitrary system of basic cells. If $\delta_{ij} \leq 0$ for every possible indices i, j then the optimal solution has been reached. If, on the other hand, for a pair $(i, j) = (i_1, j_1)$, δ_{ij} is positive, then consider the simple loop passing through the cell (i, j) and besides it only through basic cells. Denote the cells of this loop by

$$(i_1, j_1), (i_1, j_2), (i_2, j_2), (i_2, j_3), \dots, (i_{k-1}, j_k), (i_k, j_k),$$

where the first and the last cell are the same. The new system of basic cells H' can be obtained by adding to H the cell (i_1, j_1) and at the same time omitting from it the cell (i_r, j_{r+1}) to be determined by the well known way.

Now let us construct the sets C and D which represent some rows and columns of the transportation tableau, by the following algorithmic labelling procedure. First let us label the i_1 -st row and the columns of the basic cells lying in the i_1 -st row but differing from (i_r, j_{r+1}) . At the general step let us label the rows of the basic cells lying in the previously labelled columns, provided that these rows have been unlabelled till now; then let us label the columns of the basic cells lying in the rows labelled just now but differing from (i_r, j_{r+1}) , again provided that these columns have been unlabelled till now. The process terminates when either no further row or no further column can be labelled in this way. The indices of all labelled rows will form the set C and the indices of all labelled columns will form the set D . The reduced costs belonging to the new system of basic cells H' can be computed as follows:

$$\begin{aligned} \delta'_{ij} &= \delta_{ij} && \text{if } i \in C \text{ and } j \in D \\ & && \text{or } i \notin C \text{ and } j \notin D, \\ \delta'_{ij} &= \delta_{ij} - \delta_{i_1 j_1} && \text{if } i \in C \text{ and } j \notin D, \\ \delta'_{ij} &= \delta_{ij} + \delta_{i_1 j_1} && \text{if } i \notin C \text{ and } j \in D. \end{aligned}$$

The subroutine named RUN operates with using this method for the transformation of the reduced costs. The other similar transportation subroutine, named TRNS, instead of the explicit using of the reduced costs, only stores the values of the dual variables u_i , v_j and the values of the reduced costs are computed from these (on the base of the method of potentials). For the transformation of the dual variables the following formulae can be applied:

$$u'_i = \begin{cases} u_i - \delta_{i_1 j_1} & \text{if } i \in C \\ u_i & \text{if } i \notin C, \end{cases}$$

$$v'_j = \begin{cases} v_j + \delta_{i_1 j_1} & \text{if } j \in D \\ v_j & \text{if } j \notin D. \end{cases}$$

4. SUBROUTINE RUN FOR THE PRIMAL SIMPLEX METHOD APPLIED TO THE TRANSPORTATION PROBLEM

This subroutine was written in ANSI FORTRAN language. It uses only primary (main) storage. Costs and boundary values have to be given as integer type parameters and besides them all other parameters are integers.

The subroutine RUN can be used by the following way:

CALL RUN (M,N,MN,A,B,C,CS,DS,CT,DT,H1,H2,HX,F)

where

M is the number of origins,
 N is the number of destinations,
 MN should be assigned the value of M+N-1,
 A is an array of M elements for the quantities to be transported from the origins,

- B is an array of N elements for the quantities to be transported to the destinations,
- C is a two dimensional array of M·N elements for the cost matrix,
- H1 }
H2 } are arrays of M+N-1 elements (output parameters)
HX } which will give the results. They will contain the optimal basic solution in lexicographic order of the two coordinates of the basic cells, so that H1(I) and H2(I) are the indices of the I-th component of the optimal basic solution, while HX(I) is its value,
- F is the value of the objective function,
- CS }
CT } are working arrays of M elements.
- DS }
DT } are working arrays of N elements.

About the environment of subroutine RUN:

Two short auxiliary subroutines belong to it organically: SB3 and SB5. The required memory depends on the dimensions of the problem to be solved (mainly on the value of M·N).

The size limit of the solvable problems can be expressed by a limit related to the product of the number of origins and the number of destinations:

$$M \cdot N \leq 35\,000.$$

In the case of the largest solvable problems the required amount of memory is 40,5 K words.

In the course of the computations of subroutine RUN the transformation of the reduced costs is performed according to the method described in Section 3.

5. SUBROUTINE TRNS FOR THE PRIMAL SIMPLEX METHOD APPLIED TO THE TRANSPORTATION PROBLEM

This subroutine was written in ANSI FORTRAN language. The method of solution is similar to that of RUN; the most important difference between the two subroutines is that the cost matrix is stored by RUN in primary (i.e. main) storage and by TRNS in secondary (i.e. auxiliary) storage. RUN stores the matrix of the reduced costs in an explicit way, while TRNS only stores and transforms step by step the dual variables and does not store the reduced costs. Like for the subroutine RUN, also here the costs and the boundary values (i.e. the rim) have to be given as integers. With one exception all parameters of the subroutine are integers.

The subroutines TRNS can be used by the following way:

```
CALL TRNS (M,N,A,B,C, ID1, H1,H2,HX,F,OF,IER,IU,IV,CS,DS,CT,DT),
```

where

M,N,A,B,H1,H2,HX,F,CS,DS,CT,DT are the same as for the subroutine RUN (see in Section 4),

C is an array of N elements; it is the file buffer defined by the parameter ID1. It will contain one line each of the cost matrix,

ID1 is the identifier of the file containing the cost matrix,

OF is the value of the objective function in real representation,

IER is a check parameter. It returns a value different from zero, if the amounts of the two kinds of boundary values differ from each other,

IU } are working arrays of M and N elements, respectively,
IV } containing the values of the dual variables.

About the environment of subroutine TRNS: Some short auxiliary subroutines are needed: SB3 and SB5 like for RUN, and the COMPASS (assembly) language subroutines ALTERNAT, CC, LO. The file containing the cost matrix should consist of M blocks (records) with the length of 4N characters, each. This file should be filled by the rows of the cost matrix with BUFFER OUT instruction (special ANSI FORTRAN instruction on the CDC 3300 computer) before calling the subroutine TRNS.

The required size of primary memory (main storage) depends on the dimensions of the problem to be solved (mainly on the value of M+N).

The size limit of the solvable problems can be expressed by a limit related to the sum of the number of origins and the number of destinations:

$$M+N \leq 4000.$$

In the case of the largest solvable problems the required amount of main storage is 40,5K words.

In the course of the subroutine TRNS the transformation of the dual variables is carried out as described in Section 3.

6. A FINITE PRIMAL TYPE METHOD TO SOLVE THE TRANSPORTATION PROBLEM

Éva Komáromi worked out such a variant of the primal simplex method for transportation, where no cyclical recurrence can occur, i.e. this procedure always guarantees the solution of the problem in a finite number of steps. The detailed description of the procedure can be found in [2]. On the basis of this procedure the program TRAN has been written for the computer CDC 3300.

This procedure has the advantage over the conventional method that it applies the time-consuming loop-seeking algorithm only if it leads to a definite decrease of the objective

function. At the same time, if the problem to be solved is an assignment problem, then the method makes a substantially more efficient loop-seeking algorithm possible.

7. PROGRAM TRAN TO SOLVE THE TRANSPORTATION PROBLEM BY A FINITE METHOD

The program TRAN was written in ANSI FORTRAN language. Data are read from cards, while some tables of the results are carried to line printer.

The input deck of data should contain the following cards:

- a/ Control card
- characters
- 1 - 6 : number of rows,
 - 7 - 12 : number of columns,
 - 13 - 17 : blanks,
 - 18 : if its value is 1, then in the case of the indication of inconsistent boundary values the run of the problem comes to an end; if its value is other than 1, then the difference of the sums of the two kinds of boundary values are distributed among the b_j 's.
 - 19 - 23 : blanks,
 - 24 : if its value is 1, then the starting solution is generated with the northwestern corner method; if its value is 0, then the initial solution is generated by including into the basis always the cell with the least possible cost,
 - 25 - 29 : blanks,
 - 30 : in case of zero in this place, the procedure does not print the intermediate solutions, while in case of other value, it does print them,
 - 31 - 35 : blanks,
 - 36 : in case of value 1 in this place the components of the intermediate solutions are sorted for printing out according to the first subscript, in case of other value, they are not sorted,

- 37 - 41 : blanks,
42 : in case of value 1 in this place, the first set of data are followed by other sets belonging to subsequent problems to be solved; in case of other value the run of the program comes to an end after solving the actual problem,
43 - 47 : blanks,
48 : in case of zero or 1 in this place, the components of the optimal solution are sorted for printing out according to the first subscript, otherwise they are not printed in this manner,
49 - 53 : blanks,
54 : in case of value 1 in this place the components of the optimal solution are sorted for printing out according to the second subscript, otherwise they are not printed in this manner,
55 - 59 : blanks,
60 : in case of value 1 in this place, the components of the optimal solution are sorted for printing out according to their values, in a non-decreasing order, otherwise they are not printed in this manner,
61 - 65 : blanks,
66 : if it differs from zero, then the optimal basis is printed,
67 - 71 : blanks,
72 : a value different from zero indicates that the actual problem to be solved is an assignment problem.

b/ The elements of the cost matrix by rows, in format 13I6.

c/ The row boundary values (a_i) in format 13I6.

d/ The column boundary values (b_j) in format 13I6.

The result appearing on the line printer will contain the following tables:

- a/ The elements of the cost matrix by rows.
- b/ The row boundary values.
- c/ The column boundary values.
- d/ The intermediate solutions.
- e/ The optimal solution.
- f/ The number of all basis changes and within this number the number of such basis changes that result in a strict decrease of the objective function.

Maximum possible sizes of problems which can be solved by the program TRAN are

$$m \leq 100,$$

$$n \leq 100.$$

The program uses only primary storage, from which it requires 21,5 K words.

8. THE GENERALIZATION OF THE TRANSPORTATION PROBLEM FOR CONVEX OBJECTIVE FUNCTION

The objective function of the transportation problem, mentioned at the end of the Introduction, is the following:

$$\min \sum_i \sum_j \varphi_{ij}(x_{ij})$$

where $\varphi_{ij}(x_{ij})$ are piecewise linear, convex functions.

It should be mentioned, that only the demands b_j are non-negative integers, the supplies a_i are arbitrary (non-negative) real numbers. A major point of interest of the algorithm is that a particular asymmetry is introduced between the rows and columns of the cost matrix, i.e. m is far more less than n .

Applying the decomposition principle of Dantzig and Wolfe to the problem, we have a simpler problem with linear objective function and with m constraints. This will be solved with the revised simplex method, where the critical steps are:

1. How to find a feasible starting point?
2. How to generate the vectors entering the basis during the subsequent iterations?

We give some details about the latter step:

As a result of the characteristics of the original objective function, to determine whether the current solution is optimal, or which vector should enter the basis, we have to calculate

$$\min \{ \varphi_{ij}(k) - \varphi_{ij}(k-1) - \pi_i \}.$$

$$i=1, \dots, m$$

$$k=1, \dots, b_j$$

$$j=1, \dots, n$$

With j fixed, the number of differences to be scanned is $D_j + m - 1$ only. In addition, the cost matrix need be scanned only by columns, which allows us to use the internal memory even in case of a large scale transportation problem.

We had computational experiences on a CDC 3300 computer in FORTRAN. A complete redesign and rewriting of the program in SIMULA language is planned for the near future.

R E F E R E N C E S

- [1] Kéri, G.:
A modified stepping-stone algorithm for the transportation problem, Math. Operationsforsch. und Statist. 3/1972/ 327-331.
- [2] Komáromi, É.:
A finite primal method for the assignment problem, Problems of Control and Information Theory 3/1974/ 157-166.
- [3] Sz.Turchányi, P.:
Szakaszonként lineáris, konvex célfüggvényű szállítási feladat megoldása dekompozíciós módszerrel, Alkalmazott Matematika Lapok 1/1975/ 81-90.
- [4] Williams, A.C.,:
A treatment of transportation problems by decomposition, Journal Soc. Indust. Appl. Math. 10/1962/ 35-47.

A TANULMÁNYSOROZATBAN 1982-BEN MEGJELENTEK

- 130/1982 Barabás Miklós - Tőkés Szabolcs: A lézer printer képalkotás hibái és optikai korrekciójuk
- 131/1982 RG-II/KNVVT "Szisztemü upravlenija bazani dannüh i informacionnüe szisztemü" Szbornik naucsno-iszsledovatel'szkih rabot rabocsej gruppü RG-II KNVVT, Bp. 1979. T o m I.
- 132/1982 RG-II/KNVVT T o m II.
- 133/1982 RG-II KNVVT T o m III.
- 134/1982 Knuth Előd - Rónyai Lajos: Az SDLA/SET adatbázis lekérdező nyelv alapjai /orosz nyelven/
- 135/1982 Néhány feladat a tervezés-automatizálás területéről. Örmény-magyar közös cikkgyűjtemény
- 136/1982 Somló János: Forgácsoló megmunkálások folyamatainak optimalálási és irányítási problémái
- 137/1982 KGST I-15.1. Szakbizottság 1979. és 80. évi előadásai
- 138/1982 Kovács László: Számítógép-hálózati protokollok formális specifikálása és verifikálása
- 139/1982 Operációs rendszerek elmélete 7. visegrádi téli iskola

